SAPIENZA
UNIVERSITÀ DI ROMA

Adaptive and Efficient Neural Architectures: From Adaptive Computation to Interpretable AI Systems

Faculty of Computer and Automation Engineering
PhD in Data Science (XXXVIII cycle)

Alessio Devoto
ID number 1701081

Advisor
Prof. Simone Scardapane

Academic Year 2024/2025

Thesis defended on January 28th, 2026

Adaptive and Efficient Neural Architectures: From Adaptive Computation to Interpretable AI Systems

PhD thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: devoto.alessio@gmail.com

Abstract

The rapid advancement of Artificial Intelligence, particularly through deep learning and Large Language Models, has yielded unprecedented performance across diverse domains. However, this success has come at the cost of increasingly demanding computational requirements and diminished transparency. Modern AI systems rely heavily on over-parameterized architectures that, while powerful, are inefficient for deployment in resource-constrained environments and opaque in their decision-making processes, raising critical concerns for adoption in high-stakes applications. This thesis addresses the urgent need to reconcile the capabilities of state-of-the-art AI with the practical demands of real-world deployment by developing systems that are simultaneously adaptive, efficient, and interpretable. Through three interconnected research themes, we advance the foundations of AI architectures. First, we establish a comprehensive framework for adaptive computation, introducing mechanisms that enable neural networks to dynamically allocate computational resources based on input complexity and available resources. Our contributions include Adaptive Computation Modules for granular per-token efficiency, adaptive layer selection for accelerated fine-tuning of Vision Transformers, and adaptive semantic token selection strategies for edge intelligence systems. Second, we tackle critical efficiency bottlenecks in Large Language Model deployment, specifically addressing the memory constraints imposed by the Key-Value cache. We propose multiple novel compression strategies—including L_2 norm-based approaches, Q-Filters that exploit query-key geometrical relationships, and Expected Attention methods that leverage future query distributions—enabling practical deployment of LLMs with extended context windows. Third, we advance interpretable AI through mechanistic analysis of model internals and domain-specific applications of explainable AI. Our work encompasses steering knowledge selection behaviors in LLMs, analyzing residual streams under knowledge conflicts, and deploying interpretable models in high-stakes domains including high-energy physics and archaeological classification, demonstrating how transparency enables both trust and scientific insight. Collectively, this body of work demonstrates that efficiency and interpretability need not be traded against performance. By developing adaptive architectures, addressing specific deployment bottlenecks, and opening the black box of neural networks, this thesis provides both theoretical frameworks and practical solutions that move AI systems closer to widespread, trustworthy deployment across diverse real-world applications.

Contents

Chapter 1

Introduction

1.1 Adaptive, Efficient, and Interpretable Systems for future AI Development

The field of Artificial Intelligence (AI), particularly deep learning, has undergone a revolutionary transformation in the last decade. The sheer scale and power of modern neural network architectures, exemplified by Large Language Models (LLMs), have delivered unprecedented performance across a multitude of complex tasks, from natural language understanding to scientific discovery. This success, however, is tethered to increasingly demanding computational and financial costs. The prevailing paradigm often relies on **over-parameterization**—training massive, static models that, while powerful, are fundamentally inefficient and difficult to deploy in resource-constrained environments like mobile devices, edge servers, or specialized scientific instruments.

Furthermore, the complexity that fuels these gains simultaneously exacerbates the **black-box** problem. As models grow, they become less transparent, making it exceedingly difficult to understand their decision-making processes, debug failures, and establish the trust necessary for their adoption in high-stakes domains such as medicine, finance, and autonomous systems. This lack of transparency is not merely an academic concern; it raises critical ethical, legal, and safety questions.

This thesis is motivated by the urgent need to bridge the gap between the power of state-of-the-art AI and the practical constraints of real-world deployment. We assert that the next generation of AI systems must be:

1. **Adaptive:** Capable of dynamically adjusting their computational effort based on the input's complexity, the available resources, or the specific task.

2. **Efficient:** Designed to minimize computational and memory footprints during both training and inference, enabling broader accessibility and reducing environmental impact.

3. **Interpretable:** Transparent in their internal mechanisms and decision-making, providing human-understandable insights to foster trust and accelerate scientific discovery.

To address these challenges, this thesis sets forth the following central research questions:

- **RQ1:** How can the principles of **conditional computation** be generalized and formalized to create neural architectures that dynamically allocate resources at various granularities (e.g., layer, token, or module level) for both efficient training and inference?

- **RQ2:** What novel architectural and algorithmic techniques can be developed to overcome the specific memory and computational bottlenecks, such as the **Key-Value (KV) cache** in Large Language Models, to make state-of-the-art models practical for widespread deployment?

- **RQ3:** How can **eXplainable AI (XAI)** and **mechanistic interpretability** techniques be applied to both large-scale foundational models and domain-specific applications to build robust, trustworthy, and insight-generating AI systems?

1.1.1 Thesis Contributions: A high-level Overview

The body of work presented in this thesis addresses these research questions through a series of published papers, focusing on the core themes of **Adaptive Computation**, **Efficient Architectures**, and **Interpretable AI Systems**. The key contributions are summarized below and detailed in the subsequent chapters.

1.1.2 Adaptive and Efficient Architectures

The first major theme of this thesis focuses on fundamentally changing the way AI models process information to achieve dynamic efficiency. We begin by providing a foundational framework and comprehensive review of **Conditional Computation**, defining the principles by which neural networks can dynamically activate or deactivate parts of their structure based on the input [517]. Building on this, we introduce the concept of granular adaptation with **Adaptive Computation Modules (ACMs)**, a general mechanism that allows models to adapt their computational load on a per-token basis, demonstrating substantial efficiency gains for inference [609]. We also apply these adaptive principles to specific efficiency challenges, proposing **Adaptive Layer Selection** for accelerating the fine-tuning of large Vision Transformers [150], and developing an **Adaptive Semantic Token Selection** scheme for resource-aware data transmission in edge intelligence and goal-oriented communication systems [154].

1.1.3 Overcoming LLM Inference Bottlenecks

The second major contribution area addresses the unique scaling challenges presented by Large Language Models (LLMs). The memory footprint of the **Key-Value (KV) Cache** is a critical bottleneck that limits the context window and the overall deployment size of LLMs. We directly tackle this by proposing a set of novel and effective strategies for **KV Cache Compression**. Our work includes a simple yet effective L_2 **Norm-Based Strategy** [153] and two more sophisticated approaches: **Q-Filters**, which leverage the geometrical relationship between queries and keys to significantly enhance compression efficiency with minimal performance degradation [208] and **Expected Attention** that uses the future queries distribution to estimate attention in current context [151].

1.1.4 Interpretable AI Systems and Transparency

The final theme shifts from efficiency to the crucial requirement of transparency and trust. We contribute to the foundational understanding of LLMs through **mechanistic interpretability**, developing methods to **Steer Knowledge Selection Behaviors** using SAE-based Representation Engineering [678] and by rigorously **Analysing the Residual Stream** of models when they encounter knowledge conflicts [680]. This work moves us closer to models that can be debugged and controlled. Furthermore, we demonstrate the broad applicability of eXplainable AI (XAI) in high-stakes, non-traditional domains. This includes the design of **Mixture-of-Experts Graph Transformers** for interpretable particle collision detection in High-Energy Physics [200] and the **Interpretable Classification of Archaeological Data** [87], providing domain experts with new insights and validating AI decisions in scientific discovery.

1.2 Additional Relevant Publications

In addition to the core papers driving the narrative of adaptive efficiency and interpretability, the following publications further expand the scope of my research, addressing foundational aspects of trustworthy AI such as robustness and continuous learning: "Reidentification of Objects From Aerial Photos with Hybrid Siamese Neural Networks" [152], "On the Robustness of Vision Transformers for Monocular Depth Estimation" [178], "Are We Done with MMLU?" [197], "Class Incremental Learning with Cascaded Gated Classifier" [467].

1.3 List of Publications

Table 1.1. Publications List with Venue and Year

Title and Citation	Venue	Year
Are We Done with MMLU? [197]	NAACL	2025
A Simple and Effective L_2 Norm-Based Strategy for KV Cache Compression [153]	EMNLP	2024
Steering knowledge selection behaviours in LLMs via sae-based representation engineering [678]	NAACL	2025
Adaptive Semantic Token Selection for AI-native Goal-oriented Communications [154]	GLOBECOM	2024
Conditional computation in neural networks: Principles and research trends [517]	Intelligenza Artificiale	2024
Adaptive Modular Computation: Granular Conditional Computation For Efficient Inference [609]	AAAI	2025
Reidentification of Objects From Aerial Photos With Hybrid Siamese Neural Networks [152]	IEEE Transactions on Industrial Informatics	2023
On the robustness of vision transformers for in-flight monocular depth estimation [178]	Industrial Artificial Intelligence	2023
Adaptive layer selection for efficient vision transformer fine-tuning [150]	Neurocomputing	2025
Q-Filters: Leveraging QK Geometry for Efficient KV Cache Compression [208]	SLLM @ ICLR	2025
Goal-oriented Communications based on Recursive Early Exit Neural Networks [468]	Asilomar Conference	2024
Class Incremental Learning with Probability Dampening and Cascaded Gated Classifier [467]	Neurocomputing	2024
Analysing the residual stream of language models under knowledge conflicts [680]	MINT @ Neurips	2024
Expected Attention: KV Cache Compression by Estimating Attention from Future Queries Distribution [151]	arXiv preprint (under review)	2025
Universal Properties of Activation Sparsity in Modern Large Language Models [559]	arXiv preprint (under review)	2025
Interpretable Classification of Levantine Ceramic Thin Sections via Neural Networks [87]	ML: Science and Technology	2025
Attention Sinks in Diffusion Language Models [506]	arXiv preprint (under review)	2025
Mixture-of-Experts Graph Transformers for Interpretable Particle Collision Detection [200]	Nature Scientific Reports	2025

Chapter 2

Conditional Computation

In the last twenty years, neural networks (NNs) have undergone two opposing trends. On one hand, the number of practical applications has continued to grow, fueled by successes in, among others, language modeling [283], drug design [593], rendering, and language-vision reasoning [462]. On the other hand, their design has crystallized around a very small set of layers (e.g., multi-head attention) and principles (e.g., permutation equivariance), while the focus has shifted on scaling up their training, both in terms of data and parameters [283]. Apart from scale, maybe less than a dozen components and variations thereof are enough to categorize the vast majority of neural networks deployed nowadays.

Among these design principles, *sequentiality* has remained a key component. NNs, be they convolutional, recurrent, or transformers, are composed of a stack of differentiable operations, which are activated in sequence for each input to be processed. Stated in another way, their *computational graph*, i.e., the sequence of primitive operations executed on the underlying hardware, is fixed beforehand when instantiating them. Because NNs have continued to scale in depth and width, this has led to several issues in terms of computational performance and efficiency [187]. Standard techniques to make NNs more efficient only address this partially, by replacing the original network with other *static* models having fewer layers (distillation [597], pruning [300]), less precision per parameter, or by approximating each weight matrix (e.g., via low-rank factorization).

By viewing neural networks as computing systems, this behavior is counter-intuitive. Hardware components are designed based on their expected peak usage, while only executing a fraction of their resources at any given time (e.g., memory). Similarly, software libraries and operating systems are composed of millions of lines of code, only a handful of which are selected and run in a given moment, thanks to the use of branches, loops, and conditional execution. Recently, a large body of literature has flourished on embedding similar *sparse modularity* principles in the design of neural networks [465]. Based on the original idea of *conditional computation* [46], this has blossomed into a large number of practical implementations, ranging from mixture-of-experts (MoEs) [533] to dynamic mechanisms to select tokens and layers in transformers [394]. This tutorial is intended as an organized, uniform overview and entry point into this growing body of literature.

The benefits of developing networks that can dynamically adjust their computational

graph go beyond memory or time efficiency. More and more, neural networks are treated in the same way as software [292], and deploying them requires the possibility of quickly debugging their predictions [69], continually fine-tuning them on new data, and transferring parts of their knowledge from one network to the other [17], similarly to standalone software libraries. As we will see, dynamically-activated neural networks provide a principled way to improve both zero-shot transfer and generalization [471] (Sec. 2.0.3) and explainability of the models (Section 2.0.3). This aligns with the requirements of many novel applications of NNs, from scientific discovery [593] to AI-native telecommunications [631, 550]. In particular, smart semantic communication networks [550] envisioned in the so-called *beyond 6G* model necessitate networks that can flexibly adapt to bandwidth and energy constraints, while ensuring transferability and communication through separate neural modules.

For the purpose of this tutorial, we consider three flavors of dynamism: (a) neural components that can restrict their computation to a smaller subset of input tokens (dynamic input sparsity); (b) layers that can selectively activate sub-components for processing a token (dynamic width sparsity); and (c) layers that can be completely skipped during their execution (dynamic depth sparsity). As we show in Section 2.0.1, a simple mathematical formalism encompasses all three cases. Section 2.0.1 also highlights how modularity can be achieved with the addition of a small set of primitives to our networks' toolkit, namely, the possibility of sampling in a differentiable way elements from a set. We discuss in Section 2.0.1 the simplest technique to this end, the Gumbel-Softmax trick [**?** 378], and some common extensions.

We then proceed to discuss three notable implementations of these concepts in Section 2.0.2: early-exit (EE) models, mixture-of-expert (MoE) layers, and token selection mechanisms. While these models are generally discussed in separate fashions (e.g., see [522, 384] for EEs, and [657, 181] for MoEs), viewing them as specific instances of a general framework highlights many similar trends and characteristics. In fact, we argue that designing networks with dynamically activated components is not simply a matter of enhancing performance: in all these cases, the resulting networks are more apt at adapting to variable system's constraints (e.g., decreased energy usage), specialization, catastrophic forgetting, and multimodality. We build on these insights in Sections 2.0.3 and 2.0.4, where we list potential research directions for these models including adapting their computational cost and energy at inference and training time in an elastic way, zero-shot transfer, robustness, and explainability.

Relation to prior works: we do not claim to be the first to discuss these ideas, nor to present them in a general way. We simply hope to provide a simple, cohesive entry point into an extremely fascinating and growing research direction in the literature. Recently, modular deep learning [465] has been proposed as a general term for neural networks where computation is functionally decomposed into units that are sparsely activated. Compared to [465], our formalism is simpler and we focus on a smaller set of ideas: for example, we do not consider input composition methods, nor soft routing strategies, and we focus on viewing modularity as discrete sampling inside neural networks. Hence, we provide an orthogonal view to [465], to which we refer for a broader outlook on modularity and composition. We also focus on a tutorial exposition, preferring simplicity and clarity to completeness. We only describe the simplest techniques for differentiable sampling, and we refer to [426] for a larger and more in-depth introduction to this field. Additional entry points in the literature include [239] (for an overview up to 2021 of

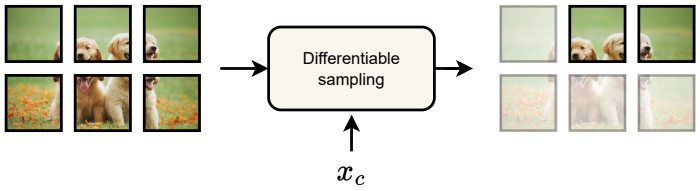


Figure 2.1. Input sparsity: A differentiable mechanism subsamples input tokens to be processed by the later parts of the network (we show original image patches in the figure, but the tokens can be equivalently be replaced by their latent representations if we consider an intermediate layer of the architecture).

dynamism in NNs) and [187] for designing faster and more efficient transformer models.

2.0.1 Three types of conditional computation

A general formalism for sparse modularity

Neural networks can be described as the composition of several trainable, differentiable operations of the form:

$$y = f(x, w) \tag{2.1}$$

where x denotes the input, w the module's parameters, and f the specific operation. Because of functional composition, f can represent an operation at any level of complexity, e.g., a basic linear algebra operation, a layer (convolutional, recurrent, ...), or a composite block, such as the classical combination of token mixing, channel mixing, and layer normalization operations found in transformers.

Most of our discussion focuses on transformer-like architectures, where the input is composed of a set of *tokens*, representing parts of the complete input [501] (e.g., subwords in a sentence or patches in an image). The input x in (2.1) is purposefully left vague: depending on the scenario and the layer, x can be a single token (e.g., for the fully-connected blocks inside a transformer), the entire set of tokens, a mini-batch of inputs, or other forms of data aggregation.

We are interested in augmenting (2.1) to allow the dependence of the different quantities to vary based on some **conditioning input** x_c, where x_c can be equal to x, some linear or non-linear projection of x (in order to decouple the conditioning input from the input used for the layer and increase the flexibility of the dynamic mechanism), or some additional input information (e.g., a latent token representing the language of the speaker in an audio model [361]).

Suppose we have available some trainable *subsampling* operation, defined over the conditioning input x_c and over some set $\mathcal{S}$, such that:

$$\Gamma(\mathcal{S}, x_c) \subseteq \mathcal{S}\,. \tag{2.2}$$

Practically, sets are always represented in some matrix format inside neural networks (e.g., by stacking all elements row-wise), in which case we assume the relevant quantities in (2.2) to follow the same conventions. As an example, if $\mathcal{S}$ is represented by a matrix $\mathbf{S}$, (2.2) can be implemented by row-wise masking of the corresponding elements:

$$\Gamma(\mathbf{S}, x_c) = \mathbf{M} \odot \mathbf{S} \tag{2.3}$$

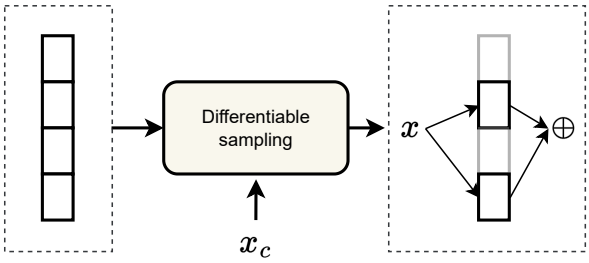


Figure 2.2. Width sparsity: Different parts of a layer (e.g., *experts*) can be activated based on the conditioning value.

where **M** is a binary mask of suitable shape with rows set to either **0** or **1**. Different implementations can be obtained depending on whether the size of the output mask is known in advance (e.g., how many tokens to keep in a given layer), or must be estimated by Γ itself. We will see later on in Section 2.0.1 how such an operation can be implemented in practice.

This small addition allows us to implement several levels of modularity inside our neural network. First, assume the input x can be decomposed into smaller components $x = \{x_1, \ldots, x_n\}$. In many cases, this is a natural decomposition, e.g., tokens inside a transformer, time instants for a sequence, or frames for a video input. However, they can also correspond to additional register tokens [132], to elements extracted from some external memory (e.g., AdaTape [637]), or additional views or modalities. Then, dynamic input sparsity (Fig. 2.1) is achieved by combining the layer with a subsampling operation on the input:

$$\text{Input sparsity} : f(\Gamma(x, x_c), w) \tag{2.4}$$

This allows the layer to focus exclusively on input components that are relevant to the current operation. As an example, consider an image with a very wide, uniformly blue background: for the majority of tasks, we can imagine the layer to be able to operate even when removing the vast majority of tokens corresponding to such background [610]. This is, indeed, observed in practice, as we'll describe in Section 2.0.2.

Second, assume the weights themselves can be decomposed into blocks $b = \{w_1, \ldots, w_m\}$, and the entire function decomposed as:

$$f(x, w) = \oplus(\{f_1(x, w_1), \ldots, f_m(x, w_m)\}), \tag{2.5}$$

where $\oplus$ is a permutation equivariant operation that aggregates the results of the different blocks. This is also a very natural decomposition in practice: for example, f_i could correspond to a single output channel in a convolutive layer, with $\oplus$ being a concatenation. Or, f_i can be a head in a multi-head attention layer, with $\oplus$ being a concatenation followed by a projection. In this case, what we call dynamic width sparsity (Fig. 2.2) can be achieved by subsampling the blocks:

$$\begin{aligned} &\text{Width sparsity} : f(x, \Gamma(w, x_c)) = \\ &\oplus(\Gamma(\{f_1(x, w_1), \ldots, f_m(x, w_m), x_c\}) \end{aligned} \tag{2.6}$$

Imposing some form of structure over w is fundamental since subsampling each weight separately would result in a highly unstructured form of dynamic pruning, which is

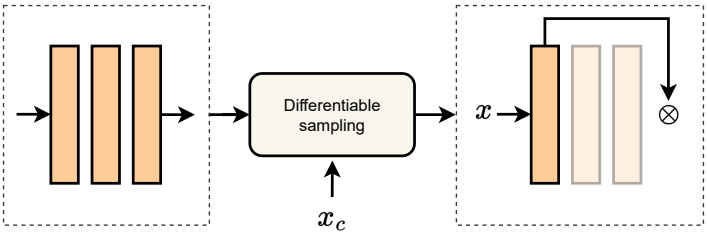


Figure 2.3. Depth sparsity: A subset of the layers can be deactivated by the sampling mechanism, like in early-exit networks.

generally very difficult to optimize and control [278]. Once again, this formulation is very generic: for example, if each f_i corresponds to an activation function, subsampling with cardinality 1 corresponds to choosing the best activation function for each token or each input, in a form of dynamic model selection mechanism [362]. If each f_i is an entire model, this can be used to route information across blocks with different types or complexity (see, e.g., layer stitching [448]). We will see in Section 2.0.2 a common implementation of this principle in the context of MoEs, and defer more general discussions to Section 2.0.3.

Third, even if a weight decomposition is unavailable, dynamic computation can still be achieved by considering the entire layer as a single block and conditionally skipping its execution. This dynamic depth sparsity (Fig. 2.3) can be achieved easily by rewriting the layer with an additional, untrainable scalar weight $\sigma = 1$, and writing:

$$\text{Depth sparsity} : f'(x, w, \Gamma(\{\sigma\}, x_c)) = \tag{2.7}$$

$$\sigma(x_c) f(x, w) + (1 - \sigma(x_c))x \tag{2.8}$$

where we assume the empty set $\emptyset$ to be implemented as 0 in practice. As a variant of this idea, we can skip entire blocks of layers by *exiting* the network instead of adding a skip connection in (2.8). We will see concrete implementations of this concept in Section 2.0.2.

Discrete sampling: the Gumbel-Softmax trick

Before proceeding, we discuss briefly the implementation of the subsampling layer Γ. As a prototypical example, we restrict our analysis to subsampling a single element from the set in input. In this case, a very common implementation is the so-called Gumbel-Softmax (GS) trick [277], also known as the concrete distribution [378]. In this section, we only provide a high-level overview, and we refer to [272] for a fuller exposition.

First of all, we process the inputs with some trainable layer $p(\mathcal{S}, x) \in \mathbb{R}^{|\mathcal{S}|}$ to provide a real-valued score for each element in $\mathcal{S}$, which is proportional to its probability of being sampled. In this section, we drop the subscript from the conditioning input x_c for readability. The implementation of p depends on the use case. For example, suppose that the conditioning input is a tensor $x \in \mathbb{R}^{h \times w \times d}$ coming from intermediate output of a CNN having width w, height h, and d channels, while $\mathcal{S}$ is the set of output channels in a convolutive layer, of which we want to select a single one [250]. We associate to

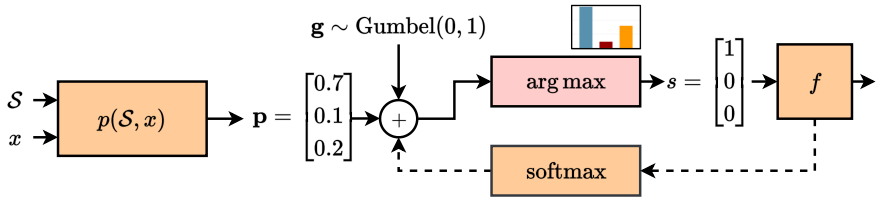


Figure 2.4. Overview of the Gumbel-Softmax trick. We show in orange differentiable operations (the argmax's gradient being zero almost everywhere) and with a dashed arrow the relaxed backward path.

each channel a trainable vector $\mathbf{c}_i$, and compute an affinity score via the dot product with a pooled representation of the image [250]:

$$\mathbf{v} = \text{MLP}\left(\sum_{i,j} x_{ij}\right) \tag{2.9}$$

$$p(\mathcal{S}, x) = \left[\mathbf{v}^\top \mathbf{c}_1, \ldots, \mathbf{v}^\top \mathbf{c}_{|\mathcal{S}|}\right] \tag{2.10}$$

Many other designs for p are possible based on x and $\mathcal{S}$. Let us denote by $\mathbf{p}$ the output of p, and by g_i samples from the so-called Gumbel distribution [277]. We also assume that the elements of $\mathcal{S}$ correspond to integers, $\mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$, in which case we simply write $\Gamma_{|\mathcal{S}|}(x)$ for the subsampling over the set of integers. Then, the following operation provides an unbiased sample from $\mathcal{S}$, with the probability of sampling the i-th element $p(s \in \mathcal{S}) \propto \exp(p_s)$ proportional to p_s:

$$s = \arg\max_i \; \{p_i + g_i\} \; . \tag{2.11}$$

Removing the Gumbel noise g_i corresponds to taking the element with the highest score p_i, while sampling provides a degree of freedom that is helpful in exploring possible alternatives. Since the probabilities p_i are implicitly trained via $p(\mathcal{S}, x)$, the network can learn to select the element from $\mathcal{S}$ which is most useful for the specific input x. In the case of channel selection, for example, this can lead to specializing single channels to specific types of inputs.

The operation in (2.11) is not easily trainable via gradient descent (since its gradient will be zero almost everywhere), but it can be relaxed with a softmax approximation:

$$s_i = \frac{\exp((p_i + g_i)/\tau)}{\sum_j \exp((p_j + g_j)/\tau)} \; . \tag{2.12}$$

The quality of the approximation can be controlled by the user-defined parameter τ, sometimes called the temperature [272]. The values in $\mathbf{s}$ are not binary anymore, being generic convex combinations of the (one-hot) representations from the elements in $\mathcal{S}$. However, a common relaxation is to use the binary values in (2.11) during the forward pass of the network, and relax the gradients to use the soft approximation in (2.12) during the backward pass. This is called straight-through estimation (STE). A high-level overview of the complete schema is given in Fig. 2.4.

The GS trick can be easily extended to sampling more than a single value by replacing the arg max with a top-k operation [307]. Suitable generalizations, such as the entmax family [118], can also sample binary vectors with a variable number of elements. The simplicity of the GS trick makes it widespread in many applications, but several other types of sampling layers can be found in the literature, especially for more complex combinatorial spaces, for which we refer to [426]. A larger overview of possible sampling operations for MoE layers is also given in Sec. 2.0.2.

2.0.2 Concrete implementations

The previous section has introduced a generic framework for designing networks with dynamic computational graphs. In the literature, these ideas have coalesced around a few, notable set of implementations. In this section we overview three of the most popular ones: **early exits** (Sec. 2.0.2), **MoE layers** (Sec. 2.0.2), and **token sampling** (Sec. 2.0.2) as examples of networks having dynamic depth, width and input, respectively. In all cases we focus on the key strengths and drawbacks of each approach, highlighting how they connect to the framework of Sec. 2.0.1.

Early exits

Early exit neural networks (EENNs) were introduced based on the idea that not all the inputs to a network are required to go through all the layers of the model to be correctly classified [565]. In fact, the accuracy of a network can decrease with respect to the depth on particularly easy samples, a phenomenon known as *overthinking* [516].

Consider a neural network (sometimes called the backbone network in this context), which can be either trained from scratch or pre-trained. In an early-exit framework, the backbone is augmented with auxiliary classifiers (early exits) which are connected at intermediate outputs of the backbone. These auxiliary blocks can be trained all at the same time or in a layer-wise fashion [522], while at inference time they are used to halt the computation when the model is confident enough about the prediction. This is an example of dynamic depth sparsity (Section 2.0.1), except that an entire subset of layers is skipped at the same time.

We introduce first the most common formulation of EENNs, in which early exits are jointly trained while the early exit logic is defined manually with additional hyper-parameters. As already stated, we can see a neural network as a composition of b sequential blocks:

$$f(x) = \mathrm{f}_b \circ \mathrm{f}_{b-1} \circ \cdots \circ \mathrm{f}_1(x) \tag{2.13}$$

where $\circ$ denotes function composition, b is the last layer of the backbone, and we remove the dependency on the parameters for simplicity. Where to place the auxiliary classifiers is in general a hyper-parameter [522], but here, for clarity, we add a classifier after each block, producing multiple prediction functions:

$$y_i(x) = \mathrm{c}_i \circ (\mathrm{f}_i \circ \mathrm{f}_{i-1} \circ \cdots \circ \mathrm{f}_1(x)) \tag{2.14}$$

where c_i is a small classifier (e.g., for a CNN, a typical design is to have a global average pooling operation followed by a fully-connected layer, while for a transformer we can apply a fully-connected layer on a specialized class token). This formulation leads to

an EENN which is capable of classifying input samples at any stage of the computation through the sequence $y_1(x), y_2(x), \ldots, y_{b-1}(x), f(x)$. Ideally, each classifier can specialize on a subset of the training samples based on the complexity of the input. In the simplest case, given a desired output y, all early exits can be trained simultaneously by minimizing the sum of the losses for each classifier:

$$L = \alpha \text{CE}(y, f(x)) + \sum_{i=1}^{b-1} \beta_i \text{CE}(y, y_i(x)), \tag{2.15}$$

where CE is the cross-entropy loss, and α, β_1, …, β_{b-1} are hyper-parameters that balance the loss contribution from each early exit. At inference time, an exit can be chosen by comparing, e.g., the predicted class probability or the entropy of the prediction to a user-defined parameter acting as a threshold. Other strategies include combining predictions based on trainable [521] or geometric [611] ensembling.

These networks have found applications in many fields recently. Concerning NLP, many strategies with BERT have been proposed [687, 628, 693], varying from patience-based EE [687] to extra classification layers [628]. For computer vision, EE applications vary based on the strategy, including multi-exiting [37, 308, 597, 64], and loss-weight adjustment [596]. Also, the tasks vary, spanning semantic segmentation [308], video recognition [204], adversarial training [294], and image compression [631]. Recently, EE strategies have been proposed also for vision-language models [562].

Although this formulation of EENNs (or minor variations thereof) is very common in the literature (see e.g., [565, 64, 687, 384, 525, 562]), it only fits partially our view of dynamic sparsity, since the computational graph is unchanged during training and the dynamism is only obtained via an external heuristic. To this end, [521] proposed a mechanism to optimize the early exit selection during training, called *differentiable branching*. Suppose we augment each early exit with an additional output that we denote as $\gamma_i(x)$. The key idea of differentiable branching is to define a new recursive output, which is given by a soft composition of the original outputs:

$$\tilde{y}_i(x) = \gamma_i(x) y_i(x) + (1 - \gamma_i(x)) \tilde{y}_{i+1}(x) \tag{2.16}$$

where the recursion stops at $i = b$, where $\gamma_i(x) = 1$ by definition. At inference time, this can be turned into an early exit module by viewing $\gamma_i(x)$ as a binary classifier and thresholding it at 0.5. By replacing the gates $\gamma_i(x)$ with samples from a GS distribution, we return to the general setting described in Section 2.0.1. In fact, if the outputs of $\gamma_i(x)$ are binary we can identify:

$$\Gamma_{b-1}(x) = \begin{bmatrix} \gamma_1(x) \\ (1-\gamma_1(x))\gamma_2(x) \\ \vdots \\ \left[\prod_{i=1}^{b-2}(1-\gamma_i(x))\right]\gamma_{b-1}(x) \end{bmatrix} \tag{2.17}$$

with the subsampling block over the indexes of the early exits. This can also be given a Bayesian interpretation under the stick-breaking process [469]. While the sampler in (2.17) is conditioned on the full input, other choices are possible. For example, in multitask learning a separate task embedding can be used to select different exits for

different tasks [664], while in the case of transformers, only the class token can be used to decide an optimal exit strategy [246]. Being able to train the choice of early exit is fundamental because it allows us to regularize it further for, e.g., better inference-accuracy trade-offs: we revisit this idea in Section 2.0.3. See [521, 384] for in-depth reviews on EENNs and [521] for a fuller description of differentiable branching.

Mixture-of-experts

Mixture-of-Experts (MoEs) models were introduced more than thirty years ago as adaptive algorithms to perform dynamic ensembles of individual machine learning algorithms, denoted as *experts* in that context [657]. Recently, MoE layers in neural networks have gained popularity, especially for LLMs and large vision modules [501]. They allow scaling the model's parameters while keeping the compute budget constant, and they offer the possibility of distributing the training by decomposing the computation of a layer over multiple GPUs [501]. As a side product, they offer dynamic width sparsity by selectively activating only parts of the layer for each input. At the time of writing, the largest open source language and vision models are based on a MoE formulation [331, 182, 501, 123, 283].

In a MoE layer, the trainable parameters are grouped into topologically identical blocks called *experts*. In the forward pass, a decision block (called *router* in this context) selects a subset of expert blocks (typically, only a fixed number of k experts, with k selected beforehand, is chosen) that will be activated and routes the input data to the selected experts. More formally, the MoE layer can expressed as:

$$f(x,w) = \sum_{i=1}^{n} \gamma_i(x) f_i(x, w_i) \tag{2.18}$$

where γ is the routing function and $f_1, f_2 \dots f_n$ are the experts. In its simplest form, this is a sparse linear combination of the outputs of the experts. This is immediately seen to be an instance of 2.6 with $\Gamma_n(x) = [\gamma_1(x), \gamma_2(x), \dots, \gamma_n(x)]$, provided that the outputs of $\gamma_i(x)$ are sparse.

The unique advantage of MoEs is that only a fraction of the network parameters, depending on the chosen value for k, are used for computation. Such fraction can be tuned by changing the value of k. Additionally, one can devise routing functions such that experts focus on different areas of the input space, leading to specialized sub-networks (see Sec. 2.0.3 for a discussion on specialization and Sec. 2.0.2 for an overview of routing algorithms for MoE layers).

As stated earlier, the seminal idea of MoEs dates back to [287], where this paradigm is regarded as an ensembling technique with soft gates. The first work that adopts MoEs in a modern deep learning scenario is [531]. Here, the gate becomes sparse, i.e., a fixed top-k operation and the main goal is scaling the number of parameters. Based on these initial results, in the past few years, the deep learning community witnessed a steep growth of MoE's success in conjunction with transformer architectures.

Unlike early exiting, where the routing function typically works at the level of input samples (e.g., images for a CNN), an MoE layer as in (2.18) can work at the level of individual tokens in a transformer. More specifically, in the fully-connected layers of a transformer block, each token can be assigned to one or more experts. Transformer

MoEs have been proposed for NLP [182, 477], for vision [501] and even for multimodal training [419]. Additional variants can be applied to individual heads of an attention block [671] or adapters in a fine-tuning context [660].

Routing in Mixture of Experts

Different routing strategies can be adopted to orchestrate the experts. These are defined in the gating function γ and usually leverage some differentiable sampling method similar to the Gumbel-Softmax trick (Sec. 2.0.1). Routing is a crucial aspect of any MoE architecture, as it must perform a matching between input data and expert network, conditioned on the input data itself. Decisions taken by the routing function govern the training. They can determine faster convergence and expert specialization or cause *expert starvation*, a peculiar situation where only a few experts are always activated due to their ever-increasing expertise.

The routing algorithm usually adopts a dot product for computing similarity between the embedding of each routed sample (e.g. a token in a transformer) and n trainable expert embeddings (one for each expert). Once this operation has occurred, the assignment can happen according to various methods.

In *top-k token choice*, each token is assigned to the k most similar experts, where k is an arbitrary value. This is the original approach from [531], and it usually requires additional regularization to avoid expert starvation and training instability. The same approach has been adopted with minor changes (e.g. balancing loss and the k factor) in [492, 182]. *Top-k expert choice*, proposed in [689], tries to solve the expert balancing problem by assigning each expert to the k most similar tokens, where k is an arbitrary value. Other works [114, 504] cast the expert assignment problem into a *reinforcement learning* framework and train over a routing policy to assign each token to a single expert. Finally, [702] achieves surprisingly good results by randomly assigning experts to routed samples. An empirical comparison of different routing functions can be found in [368], while a broader overview of MoE layers is given in [181].

Soft routing

We discuss here a variant of the MoE layer in (2.18), known as *soft* MoE, which has become popular recently to avoid training instabilities in the layer. The key idea is to use the gating function to combine the layer's inputs [476] or weights [416] instead of the model's outputs. These variants also provide a fixed compute budget which is decoupled from the total number of parameters, but they are generally easier to implement and train.

First, denote by $x_1, \ldots, x_n$ the tokens in input to the layer. The first soft MoE variant we consider can be defined as [476]:

$$f_i(x, w_i) = f_i\left(\sum_{i=1}^{n} \gamma_i(x_i) x_i, w_i\right) \tag{2.19}$$

i.e., each expert is activated with a soft combination of input tokens. Compared with standard top-k routing, each expert is only activated once in this formulation, irrespective of the number of tokens. The output of the layer for a single token is then given by

another soft combination, this time over the experts' outputs:

$$f(x_i) = \sum_{i=1}^{n} \gamma_i(x_i) f_i(x, w_i) \tag{2.20}$$

By comparison, another soft MoE variant can be achieved by performing a soft combination of the *experts' weights* [416]. Denote by $f(x, w)$ the generic architecture of a single expert, we have:

$$f(x) = f\left(x, \sum_{i=1}^{n} \gamma_i(x) w_i\right) \tag{2.21}$$

A thorough comparison of the relative benefits of hard vs. soft routing is still lacking in the literature, but we highlight a few benefits in Sec. 2.0.3.

Token dropping and token merging

We conclude with a brief analysis of token selection mechanisms. In particular, we consider two cases of token selection, which we refer to as *token dropping* (removal of tokens from the model) and *token merging* (combination of tokens). Both are conditional computation techniques that can be employed in any transformer model. The fundamental idea behind token dropping and token merging is that the input data often contains information that is nearly useless for the final task. Since transformers operate on set-based inputs with no fixed cardinality, one can dynamically reduce the number of tokens being passed to each layer depending on the relevance of said tokens for the final task.

More formally, let $\mathbf{X}$ be the set of n tokens inside a transformer model stacked horizontally. For the purpose of this section, the subsampling operation Γ can be implemented as a matrix multiplication of the input matrix $\mathbf{X}$ and a mask $\mathbf{M}$:

$$\mathbf{X}' = \Gamma_n(\mathbf{X}) = \mathbf{M}\mathbf{X} \tag{2.22}$$

Depending on the properties of the matrix $\mathbf{M}$, we can obtain either token dropping or token merging:

- Token dropping: $\mathbf{M}$ is a matrix of shape $n' \times n$ that selects a subset of n' elements from the original tokens. This can be achieved by setting each row to a one-hot vector.
- Token merging: $\mathbf{M}$ is a binary matrix of shape $n' \times n$, with the requirement that each column sum to one. In this way, the output tokens are combinations of the input ones, so each input token participates in a single output token. This can be seen as a dynamic variant of the standard pooling used in CNNs.

Both methods aim to reduce the computational cost of the forward pass of a transformer by reducing the number of tokens. The mask $\mathbf{M}$ is usually computed depending on the input data, $\mathbf{M} = \text{MLP}(\mathbf{X})$, in such a way as to eliminate only redundant information, such as background patches in an image. This is usually done either by small modules that are added to the backbone model [495] or by evaluating some channels added to the tokens for this purpose [394]. For token selection, when dealing with mini-batches of

inputs, the sub-sampling operation in (2.22) cannot be performed easily in a vectorized way. In these cases, (2.22) can be replaced by a masking operation as in (2.2), with the additional constraint that subsequent selection modules can only select values that were not masked previously [495]. In these cases, performance gain only materialize at inference time, while at training time most operations are still executed in a masked way.

Token dropping and token merging have been largely explored in recent years; some of the most impactful works on these topics in the field of vision transformers are [394, 495] for token dropping, while [65, 449] explored token merging. These techniques have also been employed in other fields where transformers are predominant: in 3D computer vision token dropping has been applied to point cloud transformers by [641], while in NLP [259] applied token dropping in the pre-training of BERT, obtaining significant speedups.

2.0.3 Research directions

Designing neural networks with sparse modularity principles has a number of benefits. The first and most studied is increased efficiency, both in training and in inference (Sec. 2.0.3). However, modular networks show emergent properties also in terms of specialization and transferability (Sec. 2.0.3), as long as providing a blueprint for a new type of explainability techniques (Sec. 2.0.3). We briefly overview these aspects in the following sections.

Efficiency in training and inference

Conditional computation has gained significant attention for accelerating training and inference in deep learning models [187, 465]. Often, the time saving comes with a drop in accuracy, making it important to assess the accuracy/computation trade-off. An interesting example is shown in Fig. 2.5. The approaches can vary based on several factors. For example, some research investigated the sparsification of CNNs [589] and of transformers [278, 102], some others focused on some specific aspects of the net, such as the gradients [147], the backpropagation, the activations [104] and the attention layers [576, 110]. For clarity, we focus here, mainly, on efficiency aspects related to the three families of models seen in Sec. 2.0.2, focusing on emerging trends and open challenges. To make the analysis more timely and precise, we have reported all the methods analyzed in Table 2.1.

When training from scratch an EENN, the joint training described in Sec. 2.0.2 cannot provide faster training since all exits must be trained simultaneously. Soft combinations of the early exits [521] combined with sampling tricks could be useful, but this aspect has been scarcely explored in the literature, possibly due to training instabilities and collapse. Still, EEs can provide indirect benefits to training by accelerating the convergence of deep neural networks [64, 565]. They can also improve the inference efficiency of large-scale pre-trained models [240, 687], making them more discriminative [565], act as a regularization technique [516], and possibly reduce training problems (e.g. vanishing gradient phenomena) [453]. In addition, they can be trained in a layerwise fashion if one starts from a pre-trained network [239]. In Table 2.1, we can see that most of these methods require additional parameters, leading to a latency reduction.

Compared to EEs, MoEs were investigated mostly to speed up training by distributing the experts across different GPUs [181], or to accelerate inference by activating a subset of experts for each forward pass. For the former, tiling the model across separate machines requires customized implementations [492]. In addition, to achieve an effective stable throughput, training must avoid collapses of the routing function, and care must be taken to balance the number of tokens that are sent to the different experts. This last point can be achieved with the addition of so-called *balancing losses*, e.g., batch prioritized routing [501, 114, 181]. Due to these problems, open-source implementations of large MoE models have generally lagged behind proprietary models, with some preliminary steps being taken in this direction [638], as observable also in Table 2.1.

The benefits of applying MoEs can also vary depending on the specific component that is being replaced. In [531], MoEs replace MLP layers to scale-up transformers. A similar methodology, for vision, was presented in [501]. Other approaches proposed to replace attention layers [671], entire blocks [561] of the net, and adapters [660]. By comparison, soft MoEs (Sec. 2.0.2) can reach similar gains as standard MoE layers while being simpler to train. Finally, the majority of MoEs are trained from scratch, but recently *moefication* has emerged as an interesting research direction, in which a pre-trained model is converted to an MoE variant by clustering the activations [482, 673] (see Table 2.1 to see the approaches based on pre-training). These variants provide different accuracy-time trade-offs based on the specific routing function. Additional emerging research trends are exploring *dynamic* variants of routing to provide a flexible inference budget instead [466, 610].

Finally, reducing the number of tokens with token selection techniques is a straightforward strategy to improve the inference time of the network. Most methods in this sense have focused on token dropping [242, 652, 65]: A-ViT adopts a token halting approach (shown also in Fig. 2.5);AdaViT [394] proposes a light-weight decision network, attached to the backbone to produce decisions on-the-fly; DynamicViT [495] proposes an attention-masking strategy to block interaction of redundant tokens; MSViT [243] proposes a conditional gating mechanism that selects the token scale for every image region; GTP-ViT [635] introduces a Graph-based Token Propagation (GTP) to propagate the information of less significant tokens. During training, token selection is generally achieved by masking the corresponding input elements (Sec. 2.0.2). Designing token selection mechanisms that can improve training time (for a fixed compute budget) is still an open challenge.

Some works have focused more on specific tasks, such as diffusion [68], long-input sequences [10], segmentation [370], or to better understand the general pattern of token dropping [242]. For token merging, PatchMerger [499] proposes a module that reduces the number of tokens by merging them between two consecutive intermediate layers [499]. Recently, a hybrid solution, Token Fusion (ToFu) [300], proposed to put together the benefits of both token pruning and token merging. Token selection can also be combined with the other methods discussed in this paper. As an example, Adaptive Computation Module (ACM) [610] is a technique that combines token dropping with a dynamic width principle, in which each token is allocated a variable width for each layer in the network.

Table 2.1. Overview of methods to improve efficiency of the models, including baseline algorithms not discussed in the text (e.g., pruning). **Parameters**: whether the method adds additional parameters. **Pre-training**: whether the method requires a pre-training phase. **Latency/FLOPs**: whether the method optimizes one of these two metrics

Method	Approach	Parameters	Pre-training	Latency	FLOPs
Dynamic Convolutions [589] (2020)	Sparsity	Yes	-	Yes	Yes
Scaling Transformers [278] (2021)	Sparsity	-	-	Yes	-
SViTE [102] (2021)	Pruning	-	-	Yes	Yes
Sparse Momentum [147] (2019)	Pruning	-	-	Yes	Yes
SparseViT [104] (2023)	Pruning	Yes	-	Yes	Yes
Sparse Transformers [110] (2019)	Factorization	Yes	-	-	Yes
Differentiable branching [521] (2020)	Early Exits	Yes	-	Yes	Yes
Adaptive Early Exits [64] (2017)	Early Exits	Yes	-	Yes	-
L2W-DEN [240] (2022)	Early Exits	Yes	-	Yes	-
PABEE [687] (2020)	Early Exits	Yes	-	Yes	-
Branchynet [565] (2016)	Early Exits	Yes	-	Yes	-
AEP [516] (2023)	Early Exits	Yes	-	Yes	Yes
BoF [453] (2020)	Early Exits	Yes	-	-	Yes
DeepSpeed-MoE [492] (2022)	MoE	Yes	Yes	Yes	Yes
Sparsely-Gated MoE [531] (2017)	MoE	Yes	-	-	Yes
V-MoE [501] (2021)	MoE	Yes	-	-	Yes
MoA [671] (2022)	MoE	Yes	-	-	Yes
SUT [561] (2023)	MoE	Yes	-	-	Yes
MoV/MoLoRA [660] (2023)	MoE	Yes	Yes	-	Yes
EMoE [482] (2023)	MoE	-	Yes	-	Yes
GMoE [334] (2022)	MoE	Yes	Yes	-	Yes
MoEfication [673]	MoE	Yes	Yes	-	Yes
SADMoE [466] (2023)	MoE	Yes	Yes	-	-
A-ViT [652] (2022)	Token Sampling	-	-	Yes	Yes
ToMe [65, 68] (2022)	Token Sampling	Yes	-	Yes	Yes
AdaViT [394] (2022)	Token Sampling	Yes	-	-	Yes
DynamicViT [495] (2021)	Token Sampling	Yes	-	Yes	Yes
MSViT [243] (2023)	Token Sampling	Yes	-	Yes	Yes
GTP-ViT [635] (2024)	Token Sampling	Yes	-	Yes	Yes
Colt5 [10] (2023)	MoE	Yes	Yes	Yes	Yes
PatchMerger [499] (2022)	Token Sampling	Yes	Yes	Yes	Yes
ToFu [300] (2024)	Token Sampling	-	Yes	Yes	Yes
ACM [610] (2023)	MoE/Token Sampling	Yes	Yes	-	Yes

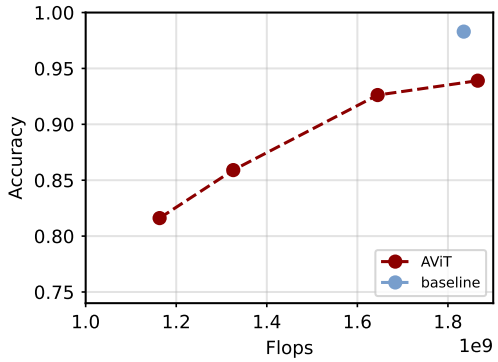


Figure 2.5. Example of accuracy-flops trade-off for inference with A-ViT [652]. Specifically, the architecture is a DeiT, trained on Imagenette.

Specialization

Modularity can also lead to specialization benefits in specific tasks [675, 406], such as language modeling [361, 400], cross-lingual transfer [463, 17, 462, 592], speech processing [400, 464], computer vision and multi-modality [493, 419, 14], and task generalization [415].

Specialization can happen in two ways: *implicitly* if no external information is provided, or *explicitly* if additional information (e.g., the speaker identity) is known. The key insight is that knowledge of, e.g., the task, the domain, or the speaker provides latent information that can be used to condition the routing blocks (Γ in Sec. 2.0.1), thus specializing specific parts of the network or specific components to different scenarios. This can be achieved in different ways: manual routing in which different components are pre-selected for the different latent vectors [465], direct conditioning of the routing functions [691], entropy regularizers to force the distributions w.r.t. to a specific latent vector to have low entropy [419], or weight sharing across tasks [562]. Soft merging [416] also seems to offer specialization benefits.

Studies on quantifying the quality of the resulting specializations were carried out recently in the literature [638]. MoEs showed promising performance due to their nature, and, specifically, hard routing facilitates module specialization [465]. On the other hand, learned routing could lead to sub-optimal results [407, 406] w.r.t to fixed routing, except for specific cases [471]. As another example, [638] showed that many routing decisions tend to ignore context, and they can be fixed during the early stages of training. Generally speaking, understanding the interplay between routing and specialization and the extent to which this specialization correlates with human-understandable semantics and biological plausibility remain open challenges that will require novel benchmarks and metrics [482, 675]. Recent moefications works show that some form of emergent modularity may also be found in pre-trained networks with no specific modularity bias [482, 673].

A benefit of having specialized sub-structures and components is that the networks can perform better in multi-task learning [415] and multi-domain scenarios [419], and they can potentially enable zero-shot transfer and generalization of the resulting sub-structures. We list a few interesting examples from the recent literature. PHAT-

GOOSE [415] is a MoE specialized in zero-shot generalization, thanks to a new post-hoc routing strategy. EMoE focuses on the implicit modular structures (Emergent Modularity) of large pre-trained transformers [482]. DSelect-k [244] presents a continuously differentiable and sparse gate for multitask learning. Uni-Perceiver-MoE [691] is a generalist conditional MoE that shows SOTA performance when compared to specialized MoEs. LIMoE [419] is focused on language-image pre-training. DeepSeekMoE [123] manages to ensure expert specialization for language models, still reducing computational costs. Other approaches include modular submodels to scale language models [52] and a class-aware channel pruning for *queriable* NNs [285].

Explainability

Finally, modularity and sparsity can provide significant gains in explainability, which is a significant issue when deploying systems [241]. In particular, analyzing and plotting the routing decisions made by the subsampling blocks Γ almost always provide valuable insights into the predictions. These include, but are not limited to, visualizing representative tokens sent to each expert [419], visualizing the early exits distribution for a given sequence in an autoregressive model [525], visualizing the tokens that were never discarded in a network having token selection [444, 394, 495], or the number of experts that were activated token-wise [610]. These techniques can provide benefit also for specific tasks, such as object detection [104].

Also in this case, we lack principled benchmarks and frameworks to analyze the resulting plots, which is an open problem in explainability in general [241]. In addition, for networks having thousands of blocks or experts, manually analyzing each of them can be a time-consuming process. LLMs can potentially help in automating this process [57]. From a mechanistic interpretability point of view, fully understanding and being able to track the evolution of modules in these networks could be a large step forward in better understanding overfitting, generalization, and fine-tuning [270]. Finally, we note that being able to plot and visualize internal decisions of the network provides a direct way for providing feedback on, e.g., whether specific modules should be kept or ignored, and to perform interventional analyses [69].

2.0.4 Conclusions and future trends

In this tutorial paper we have provided an introduction to the emerging field of designing neural networks which are *sparsely activated* in a *modular* fashion, via the use of conditional computation techniques. To this end, we have provided both a general mathematical formalism to categorize these approaches, and then an overview of several concrete implementations including mixture-of-expert models and early exit neural networks. Although these models have been investigated mostly for improving training and/or inference time, we have discussed a number of additional emerging benefits from this approach, including specialization, generalization, and explainability. Many of these benefits are only starting to be investigated, opening up interesting avenues of research.

Some common challenges have also emerged from our discussion: (a) being able to control and adapt the inference and/or training time is still an open challenge, with most techniques providing a fixed accuracy-time trade-off (e.g., MoEs with top-k

routing); (b) sparse routing introduces balancing and collapse challenges that must be taken care of; (c) most routing decisions are taken only locally (e.g., layer-wise), while taking them globally requires sampling from a combinatorially large search space requiring new sampling techniques [426]; (d) outside of accuracy, benchmarks and metrics for evaluating specialization and generalization of these models are still being developed [406].

There are also several research directions that we were not able to touch due to space constraints: these include scaling laws for modular models [114], biological plausibility [522, 675], and exploiting these models in specific applicative fields such as split computing [384] and semantic communication [664].

Chapter 3

Efficient and Adaptive Architectures for Resource-Constrained Scenarios

3.1 Adaptive Computation Modules for Granular Efficient Inference

Driven by their constantly improving capabilities, state-of-the-art neural networks have been experiencing a continued growth in size in the last decade [475]. This progress was made possible by algorithmic and model design improvements (in particular with the introduction of transformer models) and the increasing computational power of modern GPUs. Scaling up the model architecture frequently results in improved performance [149, 661], causing the size and computational costs of state-of-the-art models to keep increasing steadily. These escalating costs limit applications in latency-sensitive and energy-constrained scenarios and contribute to higher carbon emissions, exacerbating environmental concerns [526, 457].

Several approaches from the literature aim to mitigate this problem. For example, quantization reduces inference time by quantizing the weights and activations into low-precision floating-point [119] or integer values [614, 146]. Knowledge distillation methods [251, 6] can transfer knowledge from an ensemble or single larger model into a smaller network. Finally, sparsification methods [327, 238] yield either sparse weight or sparse activation tensors.

While they incur a slight decrease in downstream model performance, such methods show that some neural modules can often be replaced with computationally cheaper alternatives. Based on this observation, we argue that in transformer models [585],

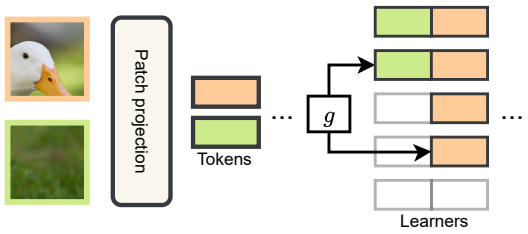


Figure 3.1. High-level overview of the Adaptive Computation Module

Figure 3.2. Example images with computation spatial load maps

the representational capacity of a whole layer is not needed for every input token. In other words, we hypothesize that the same transformation could be achieved in a more computationally efficient manner. To explore this hypothesis, we introduce *Adaptive Computation Modules* (ACMs), a neural network module that adapts its computational burden to match the difficulty of the current input token. An ACM consists of a sequence of *learners* and a single gating network. The task of each learner is to improve upon the combined output of previous learners, while the gating network determines the optimal number of learners to execute for each input token. Since all token-level decisions are independent, the resulting model is a highly granular application of the conditional computation paradigm [47, 46], and thus allows for spatial adaptability in transformer models [186, 239]. **??** provides an outline of the proposed model.

To enable the use of a vast range of publicly available models, we propose a straightforward conversion procedure in which we: (1) substitute the selected blocks of the model with ACMs, (2) initialize the learners by distilling knowledge from the substituted blocks, (3) pre-train the gating networks using artificially generated labels, and (4) train the entire model in an end-to-end manner.

Our approach can significantly decrease the model's computational footprint without sacrificing performance. We evaluate our method on the ImageNet-1k [509] dataset with Vision Transformer (ViT) models and on speech recognition with pre-trained Wav2Vec networks [34], and show that in both cases we achieve a better performance-efficiency trade-off than other conditional computation methods. We make the following contributions:

- We show that individual layers in transformer models can be computationally inefficient and that their entire expressiveness is often needed only for a small subset of input tokens.
- We introduce ACM, an easy-to-train, modular, and general-purpose module that offers a granular approach for reducing the computational cost of transformers.
- We propose a smart initialization strategy for the parameters of ACMs that relies on module-wise knowledge distillation from pre-trained models.
- We provide an efficient GPU implementation to demonstrate that ACMs effectively speed up inference.

3.1.1 Related Work

We provide a brief overview of related works, focusing on our experimental comparison. A longer overview with a broader outlook can be found in the supplementary material.

Conditional Computation

Conditional computation (CC) refers to the ability of a model to adapt its computational graph to its input. While CC can be used for problems such as continual learning [359], most CC methods adjust their execution on a per-input basis to significantly reduce their average computational cost while maintaining performance [519].

In the following, we focus on CC algorithms that can be applied to any pre-trained transformer model with minimal modifications; architecture-specific CC algorithms (e.g., dynamic channel selection for CNNs [107, 335]) are discussed in the supplementary material. We broadly categorize the methods into three groups: early-exits (EEs) [565, 64], mixture-of-experts (MoEs) [657, 531, 181], and token dropping (TD) [495, 652, 394, 242]. Finally, [78] is a concurrent conditional computation work that is remarkably similar to the proposed ACMs. In particular, it also trains a router for each layer and converts a pre-trained dense model with a three-step method.

Early-exits

In EE models, inputs are allowed to "exit" the architecture at intermediate layers via additional classifier heads that are trained together with the backbone network [565], or as a separate phase in a layerwise fashion [239], and the predictions of multiple heads can be merged with a number of different techniques [565, 611, 521]. In EE networks, the execution is stopped for the entire input at once, as opposed to individual tokens as is done in the proposed ACMs. In addition, designing and placing exit heads is itself a non-trivial problem [565], and very little work has been done outside standard computer vision and natural language processing classification tasks. In contrast, ACMs are designed to replace individual blocks in a transformer model in a plug-and-play fashion.

Token Dropping

TD strategies either prematurely stop execution [495, 652, 242] or skip computation of selected layers [394] for *individual tokens* that are deemed less relevant or redundant. We can interpret them as dynamically allocating a variable depth to each token. Our proposed ACM can be seen as an orthogonal *width* variant of token dropping, in which each token is allocated a variable *width* for each layer in the network.

Mixture-of-Experts

In MoEs, selected modules of a model are replaced by an independent set of blocks called experts, which are selectively activated for each token by an additional routing network [476]. MoEs have been successfully developed for replacing MLP layers in transformer models [531, 501], attention layers [671], entire blocks [561], and adapters [660]. Although MoEs are typically trained from scratch or fine-tuned from existing models [660], a small number of works have investigated *moefication* procedures [673, 482]. Importantly, in MoEs, each token is allocated a fixed amount of compute depending on the routing strategy (e.g., top-k routing [501]). ACM can be seen as a modification of MoEs in which the experts are ordered, and thus, the complexity of the routing problem is drastically reduced. The gating network decides *how many* learners instead

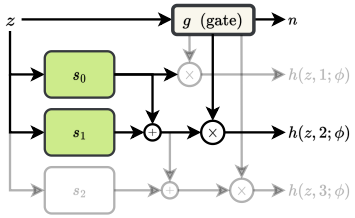


Figure 3.3. Architecture of an ACM block: the output is the sum of k *learners*, where k is determined on a per-token basis by a small gating network g. The learners are executed in parallel. In the example, only the first two learners are executed, and the computation of the third (greyed out) is skipped.

of *which* experts to execute, therefore allowing for a variable amount of computation on a per-token basis. A concurrent work of [275] defines nested experts, which results in a dynamic model similar to ACMs.

3.1.2 Method

An ACM is a conditional computation block that adapts its execution cost for each processed token via a trainable gating layer. In this paper, instead of training an ACM-based model from scratch, we focus on converting *any* pre-trained transformer network into an equivalent "ACMized" version, i.e. one having similar accuracy while achieving a pre-defined computational budget on average. To this end, we propose an effective weight pre-initialization scheme. First, we substitute a subset of layers of the base model (e.g., MLPs or MHA projections) with an ACM of similar size. The ACMs are then trained independently but *in parallel*, first with a per-layer reconstruction loss to initialize the learners and then using a cross-entropy loss to initialize the gates. The actual training consists of fine-tuning the model in an end-to-end manner to allow it to adapt its weight to the dynamic inference setting. In the supplementary material, we demonstrate that this setup significantly speeds up the training of the ACMized networks in all cases.

Adaptive Computation Module

Our ACM module aims to allow the model to work well with any required computational budget while still ensuring efficient execution on GPUs, a property most of the dynamic width methods lack [335]. Given an input token z, consider a set of N homogeneous modules, $s_n(z;\phi_{(n)})$, $n \in \{1,...,N\}$, each module having weights $\phi_{(n)}$. We refer to these modules as *learners*. In an ACM, each learner progressively refines the prediction of the previous ones such that the k-th output of the ACM block, $k \in \{1,...,N\}$, is given by:

$$h(z,k;\phi) = \sum_{n=1}^{k} s_n(z;\phi_{(n)}) \tag{3.1}$$

All intermediate outputs $h(z,1;\phi),\ldots,h(z,N;\phi)$ are valid choices for the output of the ACM: a larger value for k yields a more accurate result at the cost of a higher computational burden, while $k=1$ means that only a single learner is executed. Note that once k is known for a token, the learners can be executed in parallel.

For any token z, k should be chosen as the smallest possible value that guarantees good network performance. To this end, at the beginning of the ACM block, we add a small trainable gating network g with weights ω to select the number of learners k to be executed. In practice, the gating network returns N real-valued outputs, which are then discretized into a one-hot gating choice vector with the Gumbel-Softmax trick [277] to retain differentiability:

$$v_n = \frac{\exp((\log(g(z;\omega))_n + \gamma_n)/T)}{\sum_n \exp((\log(g(z;\omega))_n + \gamma_n)/T)}. \tag{3.2}$$

In Equation (3.2), $\gamma_1, ..., \gamma_N$ are i.i.d samples from $\texttt{Gumbel}(0,1)$, and T is softmax temperature. In the forward pass, we discretize v into a one-hot vector $\hat{v}$, but the continuous values are used directly for gradient computation in the backward pass. The complete ACM architecture is outlined in Figure 3.3.

Design of the ACM blocks Any network architecture with the required input and output dimensionality can play the role of a learner. For simplicity, in this paper, we always use two dense layers with a GELU activation function in between. Since we replace selected modules of a pre-trained model with ACMs, we always pick N and the size of a single learner such that the entire ACM module has approximately the same computational cost and the number of parameters as the substituted block. This is straightforward to achieve as the cost of a learner scales linearly with its hidden dimensionality. However, the output dimension has to be the same as in the replaced module. This results with output layer biases taking a larger share of learner parameters as we increase N. To avoid this effect, we simply eliminate them from our architecture.

For the gating network, we also use a two-layer network and determine its hidden dimensionality such that the total computational cost of gating is around 1% of the overall model cost. When feasible – such as the module being placed under a residual connection – we set the minimum number of executable learners to 0 so that even the computation of the first learner can be skipped.

ACM weight initialization

The costs of training large models are constantly increasing [551]. On the other hand, there is a large number of publicly available pre-trained models, so the capability of adapting them is a desired property for new methods. Since ACMs are designed to replace individual modules, we initially train them by distilling the knowledge from the corresponding trained modules that are being substituted. Specifically, we propose the following scheme. A trained static model $f(\cdot)$ is cloned, and selected modules (e.g., every MLP module) are replaced with ACMs in that cloned model $f_{\text{ACMized}}(\cdot)$. Each learner from every ACM is initialized randomly. For each sample x_i from a mini-batch B forwarded through the original model, we save every input token $z^l_{i,j}$ and output token $o^l_{i,j}$ of each l-th module that was replaced in the copied model. Then, every ACM is trained independently and in parallel by minimizing the mean squared error (MSE) applied for every possible choice of k:

$$\mathcal{L}(\phi) = \frac{1}{|B|SLN} \sum_{i,j,l,n} \left\| h(z^l_{i,j}, n; \phi^l) - o^l_i \right\|^2 \tag{3.3}$$

where S is token sequence length, and L is the number of substituted modules. Note that the gating network is not needed in this phase. The effectiveness of this training approach can be tested by setting a fixed k for every ACM in the model and evaluating the model on the validation set.

With learners being able to reliably imitate the replaced modules, we subsequently freeze them and train the gating networks in a similar, layer-wise approach. We frame the problem as a classification task and generate artificial labels with the following heuristic. First, we consider the outputs of all learners and compute the distance to the original output:

$$d_{i,j}^{l}(n) = \|h(z_{i,j}^{l}, n; \phi) - o_{i,j}^{l}\|_2 \tag{3.4}$$

The target label is then set as:

$$t_{i,j}^{l} = \min \left\{ n \in \{2, ..., N\} \;\middle|\; \frac{d_{i,j}^{l}(n)}{d_{i,j}^{l}(n-1)} \geq \tau \right\}, \tag{3.5}$$

where τ is a threshold hyperparameter. In other words, we select the smallest number of learners such that the relative improvement from adding one more learner is lower than the threshold τ. With these labels, the gating networks are trained using standard cross-entropy loss:

$$\mathcal{L}(\omega) = \frac{1}{|B|SLN} \sum_{i,j,l,n} -t_{i,j,n}^{l} \log(v_{i,j,n}^{l}). \tag{3.6}$$

End-to-end training

In the third phase, we finetune the entire model end-to-end to allow it to adapt its weights to the dynamic inference setting. To make the model more predictable in terms of its computational cost, we add an auxiliary loss term that penalizes for any deviation from the given target budget β_{target} on average:

$$\mathcal{L}_{\text{b}}(\theta) = \left\| \frac{1}{|B|} \sum_{i} \frac{\sum_j \sum_l k_{i,j}^{l} p^{l}}{\sum_j \sum_l N p^{l}} - \beta_{\text{target}} \right\|_1, \tag{3.7}$$

where p^{l} is the computational cost of a single learner from layer l and $\beta_{\text{target}} \in (0, 1)$ is the targeted computational budget. This term still allows for the allocation of different amounts of computational resources to samples of varying complexity and only requires to be close to β_{target} *on average*. It also affects the computational cost of future training steps, potentially accelerating the training process.

While $\mathcal{L}_{\text{b}}$ does not prevent diversity of gating choices, it does not encourage it. In practice, we find that the gating networks collapse to a state where the same number of learners is chosen for every input token, effectively turning our dynamic model into a static one. To prevent this behavior and encourage diversity, we add two additional loss terms. The first auxiliary loss term maximizes the average normalized entropy of gating choices taken for tokens in a single image:

$$\mathcal{L}_{\text{e}}(\theta) = \frac{1}{|B|L} \sum_{i,l} \frac{\sum_n a_n \log(a_n)}{\log(N)}, \tag{3.8}$$

where:

$$a^l_{i,n} = \frac{\sum_j \hat{\nu}^l_{i,j,n}}{\sum_n \sum_j \hat{\nu}^l_{i,j,n}} \tag{3.9}$$

represents a distribution between N choices in batch B for the l-th ACM aggregated for an entire sequence of tokens from sample i. The intuition behind this loss is that not every input region is equally important; hence, a non-uniform distribution of computation is required.

Finally, entire images may exhibit different difficulty levels, and enforcing diversity of gating choices for a single image at a time may not be enough. We address this with the second auxiliary loss, which is defined as:

$$\mathcal{L}_{\mathrm{d}}(\theta) = \frac{1}{|B|^2} \sum_i \sum_m \|b_i - b_m\|_1 , \tag{3.10}$$

where:

$$b_i = \frac{\sum_j \sum_l k^l_{i,j}}{\sum_j \sum_l N} \tag{3.11}$$

denotes the fraction of learners executed on sample i. It encourages the model to distribute its computational budget between easy and hard examples. The final loss that is minimized for classification tasks is given by:

$$\begin{aligned} \mathcal{L}(\theta) = & \frac{1}{|B|} \sum_i \sum_c -y_{i,c} \log(\hat{y}_{i,c}) + \alpha_{\mathrm{b}} \mathcal{L}_{\mathrm{b}}(\theta) \\ & + \alpha_{\mathrm{e}} \mathcal{L}_{\mathrm{e}}(\theta) + \alpha_{\mathrm{d}} \mathcal{L}_{\mathrm{d}}(\theta), \end{aligned} \tag{3.12}$$

where $\{(x_i, y_i)\}_{i=1}^{|B|}$ are samples from the current mini-batch B, and α_{b}, α_{e}, α_{d} are hyperparameters for weighting the different terms. In practice, we always use $\alpha_{\mathrm{b}} = 0.1$, $\alpha_{\mathrm{e}} = 0.05$, and $\alpha_{\mathrm{d}} = 0.05$. In the analysis section, we present an ablation study that demonstrates the necessity of applying the proposed auxiliary losses and shows the positive impact of their interplay. Note that the auxiliary losses are task-agnostic, allowing ACMs to be used for any other task than classification.

3.1.3 Experiments

Due to the widespread availability of pre-trained weights, we evaluate our method on popular image classification and speech recognition tasks. The aim of the evaluation is to compare the performance-efficiency trade-offs of different methods. To measure the computational efficiency, we track the number of FLOPs used by the model for each evaluated sample and report the averaged result. The source code for our experiments is available at `https://github.com/bartwojcik/adaptive_computation_modules`.

Computer Vision

We select a ViT-B [166] model pre-trained on ImageNet-1k [509] from the `torchvison` library[1] as the base model for all methods. We compare ACMized models with three

[1] `https://pytorch.org/vision/stable/models.html`

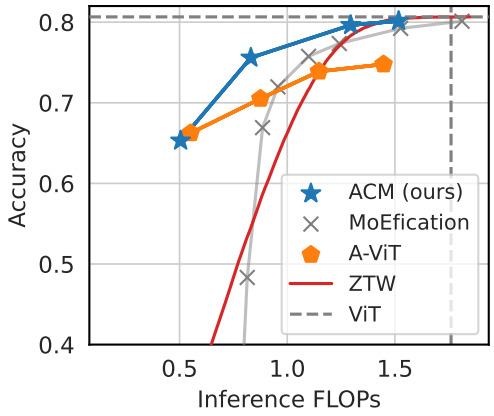


Figure 3.4. Performance-efficiency trade-offs of different conditional computation methods as measured on the ImageNet-1k dataset. ACM-based ViT-B achieves the Pareto frontier for a wide range of computational budgets.

conditional computation techniques, each one coming from a different group: A-ViT [652] (Token Dropping), MoEfication [673] (Mixture-of-Experts), and Zero Time Waste [611] (Early Exiting). For the sake of a fair comparison, we assume the same training data budget of 100 epochs for every method.

Since MoEfication requires models using ReLU activation functions, we first replace GELU activations with ReLUs and finetune the model for 80 epochs, as described by [673]. The remaining 20 epochs are used for training the routers. For Zero Time Waste, we train the early-exit heads for 95 epochs and then train the ensembles for 5 epochs. For the ACMized ViT, we replace every MLP module and every linear projection in each MHA with an ACM module (the self-attention mechanism itself is not affected). Module-wise representation distillation is performed for 2 epochs, followed by 1 epoch for pre-training of the gating networks and 97 epochs of end-to-end finetuning of the entire model. We set the number of learners N to 4 in every ACM. While MoEfication and Zero Time Waste can be evaluated with different numbers of selected experts and early exit confidence thresholds, A-ViT and ACMs need to be trained multiple times with different values of hyperparameters that approximately determine the final average computational budget. We emphasize that our A-ViT implementation includes fixes for two issues raised by GitHub users[2][3], which may affect the final performance of A-ViT. The authors of A-ViT have not yet addressed them at the time of writing this article.

We present the results in Figure 3.4. As projections in MHA are responsible for around 30% of the model cost, MoEfication seems to be hindered by the fact that it reduces only the cost of the MLP layers. A-ViT shows a gap in performance in relation to the pre-trained model, while Zero Time Waste is competitive only for higher computational budgets. ACMized ViT displays favorable performance for a wide range of computational budgets, and its advantage is especially significant below 12.5 GFLOPs.

[2] https://github.com/NVlabs/A-ViT/issues/4
[3] https://github.com/NVlabs/A-ViT/issues/13

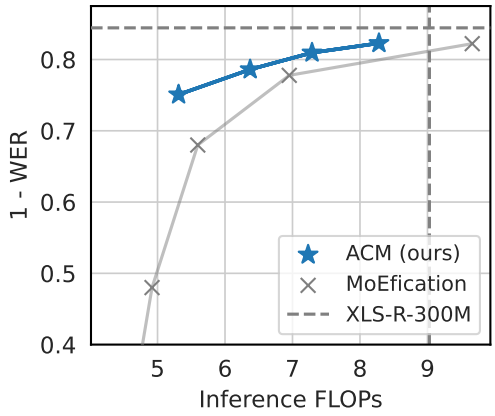


Figure 3.5. Performance-efficiency trade-offs of different conditional computation methods as measured on the CommonVoice-es dataset. The model's performance is reported in terms of Word Error Rate (WER). ACMs achieve lower WER for every evaluated computational budget.

Speech-to-Text

For speech recognition tasks, we use the XLS-R-300M [32], a pre-trained Wav2Vec2 model [34], fine-tuned on selected languages from the CommonVoice dataset [21]. Speech-to-text models introduce additional complexities for dynamic computation methods, as each token is decoded individually. Since A-ViT and Zero Time Waste were not designed for this setting, we restrict our baseline methods to MoEfication, the only task-agnostic method among those three.

We assume an equal training data budget of 10 epochs for all approaches. In the case of MoEfication, we substitute GELUs with ReLUs and train for eight epochs, followed by two epochs of router training. For ACMs, we replace every MLP block within the model with an ACM module with $N = 4$. We subsequently perform five epochs of module-wise distillation, one of training the gating networks, and four epochs of end-to-end finetuning. We present results in Figure 3.5. We see that even if only the MLP blocks are ACMized, our method still obtains a better performance-efficiency trade-off than MoEfication.

3.1.4 Analysis

Dynamic models allocate variable amounts of compute for different samples or input regions. In this section, we provide motivation for our method, examine the distribution of computational load, and show that the introduced auxiliary losses are needed. In the supplementary material, we show the impact of the proposed pre-training stages and investigate the effect of changing the number of learners N.

Computational Inefficiency of Static Modules

To justify the use of ACMs, we make the following experiment. First, we perform module-wise distillation from a selected static module of an ImageNet-1k pre-trained

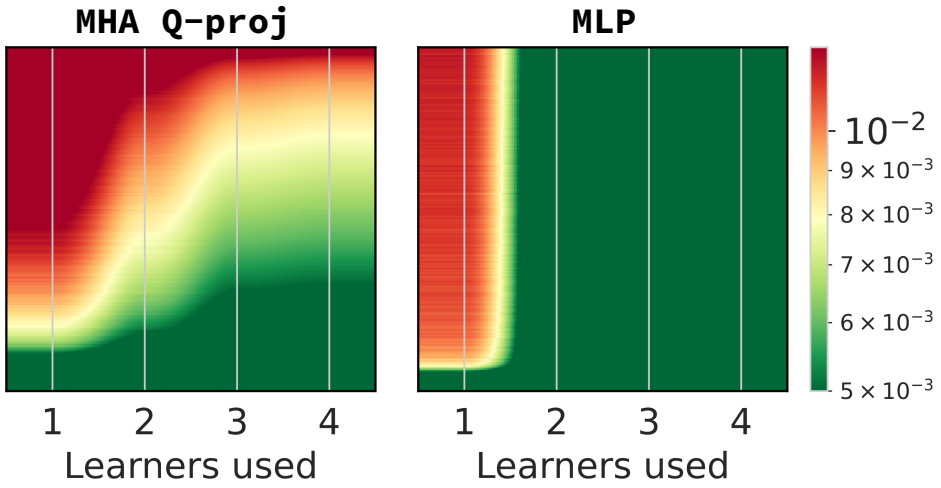


Figure 3.6. Color-coded errors of a 4-learner ACM plotted after performing module-wise representation distillation for modules from eight block of a ViT-B pre-trained model. Tokens are sorted along the Y-axis of the plot by their average error. For most input tokens, the same transformation can be performed by a considerably smaller module consisting of only two or three learners, thus justifying the use of ACMs.

ViT-B into 4 learners, just as we describe in the Method section. After convergence, we randomly select 5000 sample images from the validation dataset and forward them through the original model to save every input token $z_{i,j}$ and output token $o_{i,j}$. For every possible number of learners used $n \in \{1, ..., N\}$, we are interested in how well the learners imitate the output of the original module. For this, as in training, we use MSE: $\left\| h(z_{i,j}^l, n; \phi^l) - o_i^l \right\|^2$. The resulting tensor of MSE values has size $(5000, 197, 4)$, where 197 is the sequence length specific to ViT-B with patch size set to 16. Since ACMs process each token independently, we flatten the first two dimensions and then sort the tokens by the average error for readability. Figure 3.6 shows the resulting color-coded error visualizations for selected modules. We emphasize that the four learners have approximately the same cost and number of parameters as the original static module being substituted.

The results show that 1) only a small subset of tokens require the full processing power of the entire module, and for a majority of tokens, and 2) tokens usually exhibit varying levels of difficulty, thus warranting the use of conditional computation on a per-token basis.

Qualitative Analysis

A dynamical model should take advantage of the fact that images exhibit different difficulty levels for classification. By extracting the gating choices being done by every ACM for each patch, we can plot a heatmap that indicates which regions the model was focused the most on. This map should correlate with the meaningful regions of the image. Figure 3.7 shows that the model indeed learns to allocate its computational budget to those regions. We show that this effect is not exclusive to vision by performing a similar analysis for audio in Figure 3.8. We provide additional examples in the supplementary

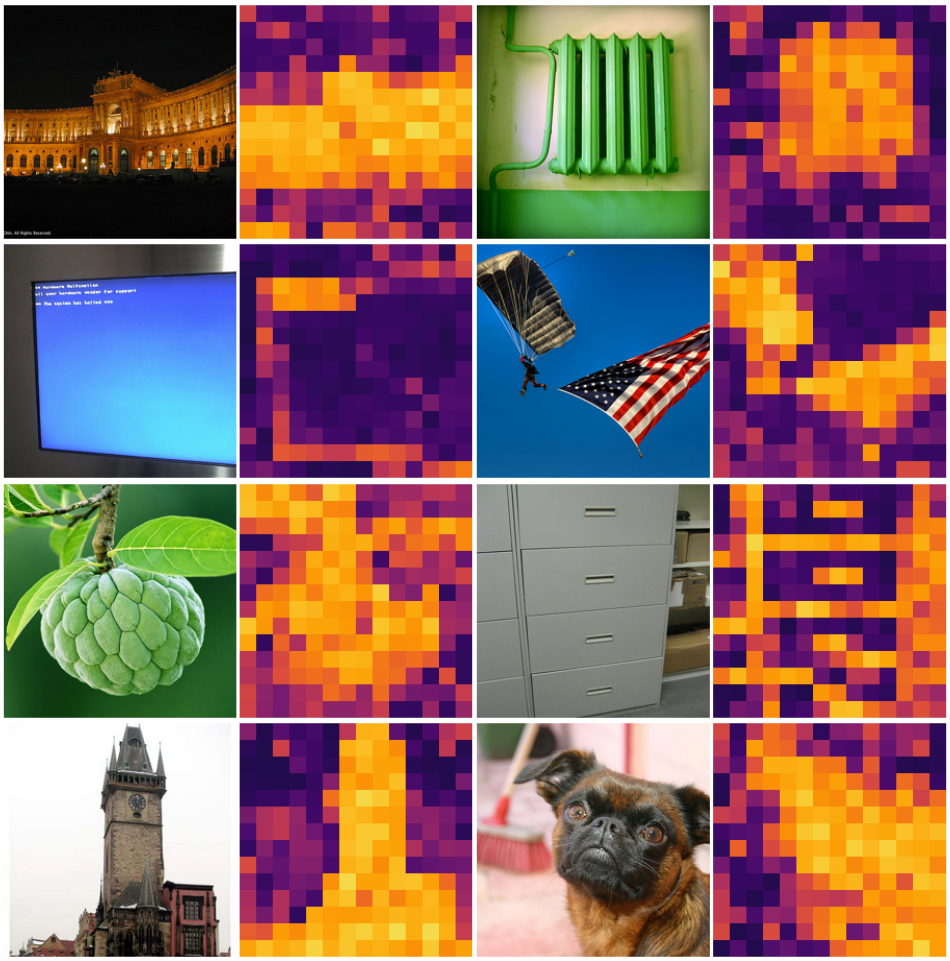


Figure 3.7. Computational load heatmaps for the model trained with $\beta_{\text{target}} = 0.6$. The model allocates its computational budget to semantically meaningful patches.

material.

Ablation Study

Being able to dynamically allocate computation when solving a task is the core feature of ACMized models. The auxiliary losses are necessary for the dynamism to emerge. We empirically demonstrate this in Figure 3.9 by training multiple runs differing only in the loss terms applied during the end-to-end training phase. For each run, we visually analyze their intra-sample computational load distribution and generate a histogram of the total computation spent on every sample.

As expected, ACMs may converge to a solution in which always the same learners are used when only $\mathcal{L}_{\text{b}}$ is applied. The inclusion of $\mathcal{L}_{\text{e}}$ helps in diversifying between different regions of the input, but overall the computation spent on each image is mostly the same. Applying $\mathcal{L}_{\text{d}}$ instead of $\mathcal{L}_{\text{e}}$ diversifies the distribution of compute spent on every image, but the gating choices for different regions of the input are not diverse on its own. Only by enabling all the proposed losses can we achieve the desired effect of

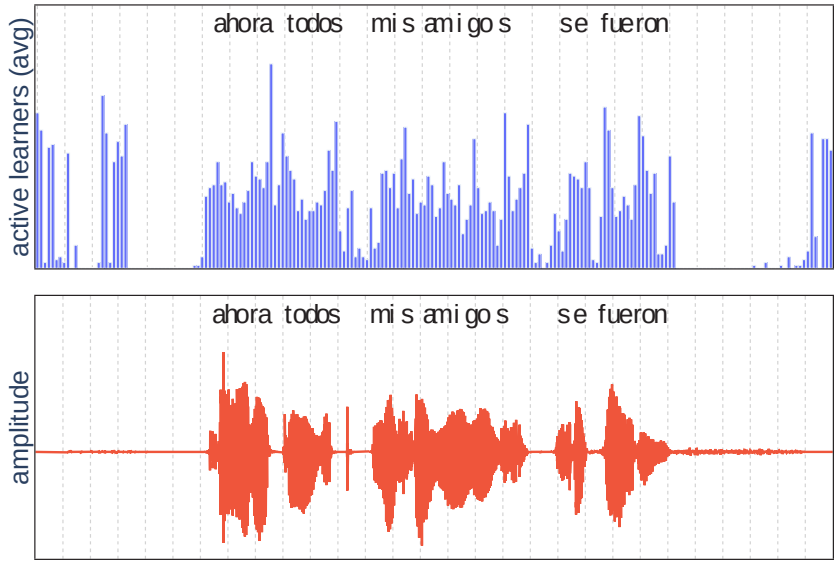
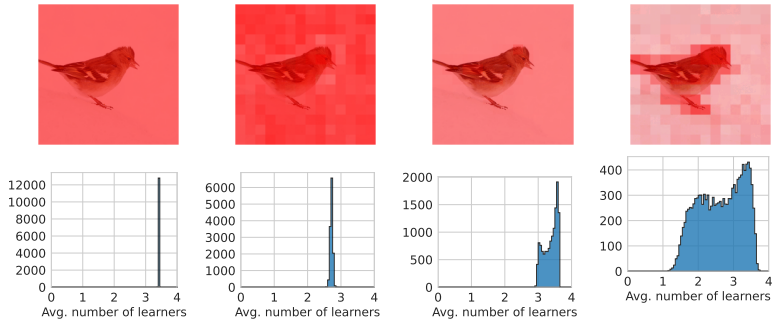


Figure 3.8. For each input audio token (red waveforms), we show the average number of learners that were activated in the ACMized model for $\beta_{\text{target}} = 0.25$ (blue bars). We can see that this model also learned to allocate its computational budget to important regions of the input.



(a) Only $\mathcal{L}_{\text{b}}$ **(b)** $\mathcal{L}_{\text{b}}$ and $\mathcal{L}_{\text{e}}$ **(c)** $\mathcal{L}_{\text{b}}$ and $\mathcal{L}_{\text{d}}$ **(d)** $\mathcal{L}_{\text{b}}$, $\mathcal{L}_{\text{e}}$ and $\mathcal{L}_{\text{d}}$

Figure 3.9. Effects of training with different combinations of enabled auxiliary losses. The computational load heatmap is rendered over the original image (top row). The histograms (bottom row) show the distribution of the total computational cost spent by the model on images from the validation set. Only combining all the proposed terms provides the desired diversity of the allocated computational budget.

the model focusing on semantically meaningful patches and having a high variability of computational budget allocation.

Hardware speedup

Due to their design, ACMs are inherently well-suited for parallel execution on GPUs. Specifically: (1) once gating decisions are determined, the execution of learners can occur in parallel; (2) tokens can be reordered without any additional copying when

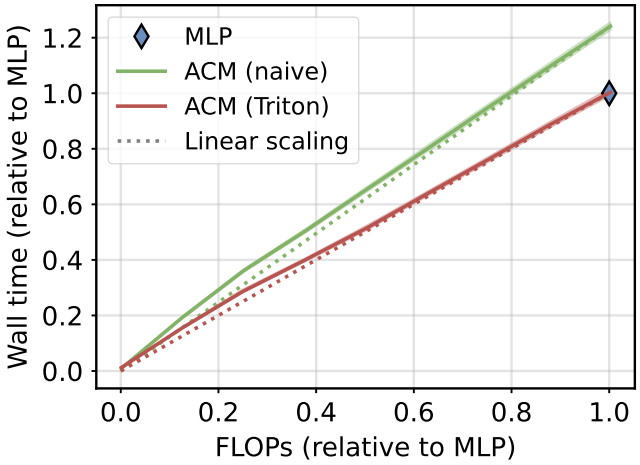


Figure 3.10. Latency of a 128-sample batch processed by a single ACM layer on an A100 GPU. The gating network is included in the measurements, but we replace the gating choices with samples from a random distribution to achieve the desired average number of executed learners. ACMs have negligible overhead, and latency scales linearly with the average number of executed learners.

they are loaded from or stored to the main GPU memory by the kernel; (3) when tokens are sorted by the number of selected learners, the computation for each group is standard matrix multiplication for which GPUs are highly optimized for; (4) the hidden dimensionality of each learner is deliberately large to maximize GPU utilization. We implement the ACM forward pass with GPU kernels written in Triton [569] and employ several optimizations including configuration auto-tuning and kernel fusion.

In Figure 3.10 we evaluate our implementation by measuring the wall-clock time of execution of a single ACM module from the ViT-B-based model. The overhead in respect to a static MLP is negligible. Moreover, the wall clock time decreases linearly with the number of executed learners.

3.1.5 Conclusion

In this work, we have demonstrated that large static modules lead to ineffective allocation of computational budget. The introduced ACM, a generic module that facilitates granular conditional computation, is designed for computational effectiveness, and models based on it achieve state-of-the-art results among the tested dynamic inference methods. Our training procedure distills a static network into an adaptive one, forcing it to allocate its computational budget to semantically meaningful input regions. Future work might explore the relationship between ACMs and pruning methods. Since our training procedure replaces individual modules with ACMs, one could also consider using quantized learners for further efficiency gains.

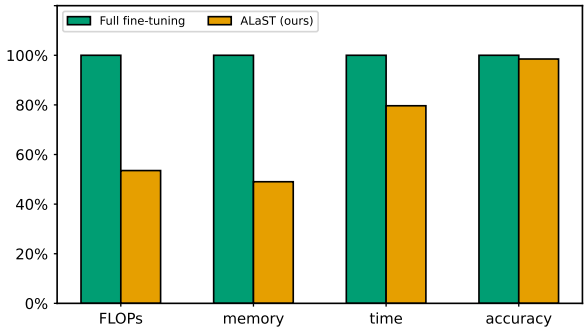


Figure 3.11. Improvement when fine-tuning with ALaST with respect to full fine-tuning for FLOPs, Memory, Wall-clock time and Accuracy. We achieve similar accuracy, with 60% FLOPs, 50% memory and 80% time reductions, averaged over all the datasets for DeiT-S. See experimental results in Section 5.4.3 for further details.

3.2 Adaptive Layer and Token Selection for Efficient Fine-Tuning of Vision Transformers

Recently, large-scale Vision Transformers (ViTs, [167]) have become the leading paradigm in computer vision. By leveraging the self-attention mechanism, ViTs can capture long-range dependencies in images at every layer, leading to superior results compared to traditional convolutional neural networks (CNNs). ViTs are at the core of a wide array of applications, ranging from resource-constrained embedded devices [76, 77, 392, 324] to large vision-language models [488, 386, 363]. However, training large-scale ViTs from scratch is unfeasible in most applications, while training from scratch at a smaller scale does not provide significant benefits compared to traditional architectures [167, 100]. Hence, a common practice is to leverage pre-trained foundation models and then perform fine-tuning for specific tasks.

In the simplest fine-tuning approach, the pre-trained ViT is frozen while only the final layers are trained on the new task. More generally, a range of alternative fine-tuning strategies—collectively known as Parameter-Efficient Fine-Tuning (PEFT) methods – such as LoRA [266], adapters [260, 558, 669], and prefix-tuning [344], have been developed to perform fine-tuning while targeting a reduced number of trainable parameters. [630].

However, PEFT methods do not significantly reduce the overall computational burden and can result in less robust and lower-quality performance, especially for smaller models running on embedded devices [537, 515]. Moreover, these methods introduce additional complexity: determining which parameters to update—even within the PEFT framework—is far from trivial. Identifying the optimal configuration often requires extensive experimentation, which may be infeasible in real-world settings. This is the case for mobile and edge devices, drones or next-generation networks that face strict constraints on memory and computational resources. These devices require lightweight models that can be quickly fine-tuned on a low budget without compromising performance. Similarly, in situations where privacy concerns prevent data from being sent to a server, on-device processing and fine-tuning become essential [76, 77, 83]. In this work, we propose a simple and effective method to accelerate the fine-tuning of ViTs in low-resource settings by leveraging two key insights. First, not all *tokens* are equally

useful during training, due to redundant information present in input images. Second, not all *layers* are equally important during training.

Building on these observations, we propose ***A**daptive **LA**yer **S**election for ViT Fine-**T**uning* (ALaST) — during fine-tuning, we allocate a scalar importance value (that we call *budget*) to each layer, which varies over time. Our method controls resource consumption along two axes: the number of discarded tokens and the selection of trainable layers, leading to the same results as a full-tuning training while being less resource-demanding (Figure 3.11). Discarding tokens significantly reduces FLOPs – due to the quadratic cost of multi-head attention – and accelerates training, especially on consumer grade GPUs, enabling models to reach higher accuracy in a shorter time. Freezing layers enhances energy efficiency and substantially reduces memory usage, which is critical for on-device training.

Our method dynamically allocates compute budgets to transformer layers based on their relevance to the current mini-batch, employing layer freezing and token reduction strategies for less critical layers. The proposed approach achieves significant efficiency improvements over traditional full fine-tuning: up to 1.5× reduction in training time, 2× reduction in FLOPs, and 2× reduction in memory usage, while maintaining competitive performance. Furthermore, ALaST demonstrates compatibility with existing parameter-efficient methods such as LoRA, making it a practical solution for deploying Vision Transformers in resource-constrained environments including edge computing and low-energy applications.

We are the first, to the best of our knowledge, to introduce an adaptive framework that systematically optimizes both token and layer selection during fine-tuning, addressing the computational challenges of training ViTs in resource-constrained environments. Unlike previous methods that perform selection statically [515], we devise a *dynamic* layer and token selection strategy. We validate our method against a comprehensive set of baseline approaches, ensuring that our results are robust and demonstrating that ALaST achieves superior efficiency while maintaining competitive accuracy. In addition, we show that our approach is orthogonal to standard PEFT methods, allowing for the possibility to combine them to further improve efficiency.

3.2.1 Related Work

Parameter Efficient Fine-Tuning

In the last years, the field of parameter efficient fine-tuning (PEFT) has seen increased interest. One of the first approaches in PEFT was the use of adapters, [260], that insert and fine-tune small neural network modules into each layer of a pre-trained model. Prefix-tuning [345] and prompt-tuning [332] learn soft prompts that are added to the input tokens. Perhaps the most popular PEFT approach is LoRA (Low-Rank Adaptation), proposed in [266], that reduces the number of additional parameters that need to be fine-tuned by decomposing the weight updates into a product of two low-rank matrices. Most of the PEFT methods were introduced for fine-tuning large language models, and subsequently applied to computer vision with varying degrees of success [630]. Typically, these approaches emphasize optimizing the number of additional parameters rather than FLOPs or training time. However, parameters memory is only a fraction of the memory used during training. Unlike PEFT, ALaST does not add any parameters, and

also reduces FLOPs and memory. Additionally, ALaST can be combined with LoRA, as we show in Section 3.2.6.

On-device Training

Several works optimize resources for on-device training. Among these, [77] was the first to highlight that memory is a major bottleneck in CNN training and devise a way to reduce memory load. Along the same line, some approaches were proposed to recompute activations for a subset of the layers to reduce memory consumption [103, 224]. While this strategy can be useful for low memory budgets, it sacrifices the compute (FLOPs), and is not suitable for devices with limited computational resources. Recently, [515, 513] explored smart initialization by identifying the best subset of layers to fine-tune. [515] identifies the most important layers and discards redundant tokens during fine-tuning. More precisely, [513] uses the relative magnitude of outputs to estimate the importance of different blocks and initialize a smaller model from a larger one. Unlike these methods, ALaST learns the importance of individual layers during fine-tuning and therefore does not need in advance extensive analyses on which layers to train.

Efficient Vision Transformers

A substantial body of research has focused on enhancing the efficiency of ViTs, particularly during the inference phase. Techniques such as model pruning, quantization, and knowledge distillation have been extensively studied to reduce the size and computational requirements of ViTs. For example, [66] proposes a token merging strategy to reduce the amount of tokens in the computational graph, while [495, 394] propose token halting, leveraging conditional computation methods [520]. [571] employed distillation from CNNs to maintain model accuracy while reducing the number of parameters and FLOPs. [360] introduced a quantized version of the Vision Transformer to decrease model inference complexity. More recently, [618] proposed to control the computational load by activating sub-modules of the network based on input difficulty. In contrast to these methods, we do not focus on inference, but address the challenge of fine-tuning available ViT models when resources are limited.

Adaptive Vision Transformers

Recent research has explored adaptive mechanisms in Vision Transformers that dynamically adjust computation based on input characteristics or training requirements. [651] introduced A-ViT, which uses confidence-based early exiting to terminate computation at different depths depending on the complexity of the input sample. [445] proposed IA-RED that adaptively reduces token redundancy by learning to identify and remove less informative tokens throughout the network layers. Another approach, [346], introduced Skip-ViT, which learns to skip transformer blocks based on input difficulty, allowing easier samples to use fewer layers while maintaining performance on challenging inputs. While these methods achieve efficiency gains during inference, they typically require architectural modifications or additional training overhead. Unlike these adaptive inference methods, ALaST provides a training-time adaptation mechanism that learns layer

importance dynamically without changing the underlying ViT architecture, focusing on training efficiency rather than inference optimization.

3.2.2 Background

Vision Transformers

A Vision Transformer (ViT) processes an image $X \in \mathbb{R}^{C\times H\times W}$ (where C, H, W are channels, width, and height respectively) through a series of L transformer layers. We formalize the transformer architecture as follows:

$$y = \mathcal{C} \circ \mathcal{F}^L \circ \mathcal{F}^{L-1} \circ \cdots \circ \mathcal{F}^1 \circ \mathcal{E}(X), \tag{3.13}$$

where $\mathcal{E}(\cdot)$ denotes the encoding network, $\mathcal{F}^i$ represents the i-th transformer layer, and $\mathcal{C}(\cdot)$ is the classification head, and $\circ$ is the function composition. The encoding network $\mathcal{E}(\cdot)$ splits the image into smaller, non-overlapping patches. Each patch is then flattened and linearly projected into a lower-dimensional embedding space, forming one token. Suppose the image is divided into N patches, each of size $P \times P$, resulting in a sequence of N tokens. This can be represented as:

$$\mathcal{E}(X) = [t_1, t_2, \ldots, t_N], \tag{3.14}$$

where $t_i \in \mathbb{R}^E$ is the embedding of the i-th patch, and E is the embedding dimension. To enable classification, a special trainable token, known as the *class token* (CLS token), is prepended to the sequence of patch embeddings. Additionally, since transformers lack an inherent sense of order, trainable positional encodings are added to each token to retain spatial information. The transformer layers $\mathcal{F}(\cdot)$ process this sequence of tokens through a series of operations involving multi-head self-attention and feed-forward MLPs. A generic transformer block at layer l transforms each token from layer $l-1$ via:

$$t_l = \mathcal{F}^l(t_{l-1}), \tag{3.15}$$

where t_{l-1} denotes the token embedding at layer $l-1$, t_l the updated token embedding, and $\mathcal{F}$ represents a standard transformer encoder block with multi-head attention and a feed-forward MLP. The self-attention operation has a quadratic complexity with respect to the sequence length, i.e., it has a cost of $\mathcal{O}(N^2)$, where N is the number of tokens [585]. This quadratic cost can be significant for devices with limited resources. Efficient implementation techniques or approximations are often required to make ViTs faster and more memory efficient for inference on downstream tasks, especially in low resource settings [394, 653, 66]. In this work, we target attention complexity by reducing the amount of processed tokens at each layer, thus speeding up the computation.

The CLS token, which aggregates information from all patches, is extracted after the final transformer layer and fed to the classification head $\mathcal{C}(\cdot)$. This head typically consists of a linear layer followed by a softmax function to produce the final classification output.

Layer Contributions during Fine-tuning

We analyze the transformer architecture from the perspective of the residual stream [175], where token embeddings flow through the model and are progressively updated

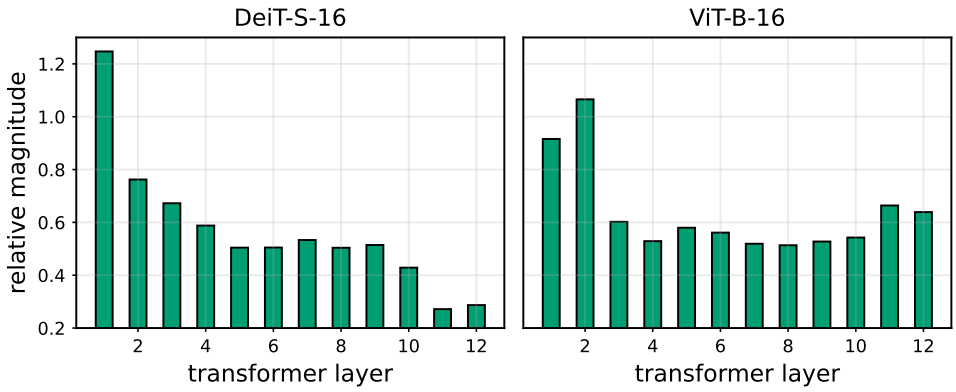


Figure 3.12. Relative magnitude $\frac{|\mathcal{F}^i(x)|}{|\mathcal{F}^i(x)+x|}$ for each transformer layer $\mathcal{F}^i$ in pre-trained DeiT-S (left) and ViT-B (right). Layers with low relative magnitudes (final ones for DeiT-S and middle ones for ViT-B) provide a minimal contribution to the residual token stream, working as identity functions.

via vector additions from attention and feed-forward blocks in each layer. This framework enables us to isolate and examine the individual contributions that each layer makes to the residual stream (Figure 3.12).

Recent studies [513, 663, 223] have revealed that layers contribute unequally to residual stream updates. This phenomenon is particularly pronounced in pre-trained models, where certain layers function almost as identity mappings, providing minimal updates to token embeddings. To quantify this behavior, we define the relative magnitude of a transformer layer as the ratio of the non-residual block's contribution to the overall layer output. Specifically, for a generic layer i, given a layer input x and the layer's contribution $\mathcal{F}^i(x)$, we follow [513] and measure the relative magnitude $\frac{|\mathcal{F}^i(x)|}{|\mathcal{F}^i(x)+x|}$ for each transformer layer. Lower values indicate layers that leave input embeddings largely unchanged. Figure 3.12 illustrates this phenomenon for both ViT-B and DeiT-S architectures.

From an efficiency standpoint, dedicating computational resources to layers that make minimal changes to input embeddings is highly inefficient. This insight has inspired optimization approaches such as distilling large language models into smaller versions by focusing only on important layers [223] or initializing smaller transformers for fine-tuning [513]. More recently, [515] identified and selectively fine-tuned the most important layers in ViTs. While these strategies yield competitive results, they typically require extensive preliminary experimentation to determine which layers to prioritize. Additionally, pre-selecting which layers to fine-tune often leads to sub-optimal results, as the most critical layers – identified through extensive search procedures [513, 515, 223] – are often highly specific to the dataset and model used, resulting in inconsistent performance across different data distributions or model architectures. For instance, an exhaustive search might identify layers 4, 5, and 6 as most important for fine-tuning on the Flower-102 dataset. Yet, this configuration may not transfer effectively to a different dataset, requiring a new round of costly search. Likewise, the most impactful layers in ViT-B may differ from those in DeiT-T, necessitating a separate search for each model. Our approach addresses this limitation by introducing a method

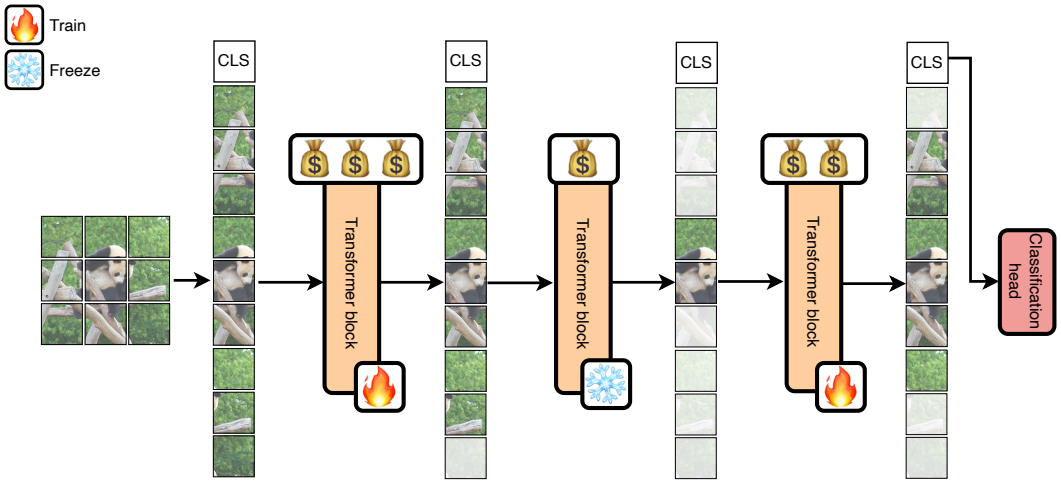


Figure 3.13. At each fine-tuning iteration, each layer is assigned a scalar importance (a *budget*). Based on the budget, we allow more (high budget) or fewer (low budget) tokens to flow through the layer - greyed out tokens are excluded from computation. Additionally, we freeze layers with the lowest budgets to save computing resources and memory.

that *automatically* and *adaptively* identifies the most influential layers during training, eliminating the need for such resource-intensive experimentation.

3.2.3 Method: Adaptive Layer Selective Fine-tuning

To overcome the challenges introduced in the previous section, we propose a simple strategy, **ALaST** (*Adaptive Layer Selection for ViT Fine-Tuning*). ALaST *adaptively* estimates the importance of each layer during fine-tuning, leading to improvements in memory usage, FLOPs, and wall-clock training time. Unlike other methods, our approach does not require any extensive experimentation prior to fine-tuning.

To save computational resources, we focus on two key parameters: the number of tokens processed by each layer and which layers are actively trained. Adjusting these parameters directly influences computational load and memory consumption. For the first parameter, we follow [394] and [495] and selectively remove redundant tokens during the forward pass. For the second parameter, we freeze less critical layers, saving memory and preventing unnecessary weight updates during the backward pass (Figure 3.13).

We now illustrate the proposed method by introducing how we estimate the value of what we call "budget" b_l for each transformer layer l. These budgets are scalar values that reflect how much compute we want to spend on each layer during training. We use them to decide, on the fly, which layers to freeze (Section 3.2.3) and which tokens to keep (Section 3.2.3) while fine-tuning.

Freezing the layers

We assume that each fine-tuning step (e.g., a batch of samples) comprises a forward and a backward pass. We use i to indicate the current step and CLS_l^i to indicate the embedding of the class token at layer l for step i. Given a fine-tuning step i, we

estimate the importance of each layer l and assign it a scalar training budget $b_l^i \in (0,1)$, representing the computational resources we can afford to spend on that layer. Notice that the budget depends both on the training step i and the layer l. The budget should be high if the layer's contribution to the final prediction is high, low otherwise. To compute the budget, we rely on the class token (CLS token) latent representation at each layer. During the forward pass, the class token aggregates information from all other tokens and it is finally fed to the classification head. Therefore, the CLS token must capture semantic information about the whole input image, making it a reliable indicator for evaluating each layer's contribution to the final prediction [352, 490]. In Figure 3.14, we empirically validate that layers with higher class token deltas exhibit larger gradient norms, confirming that our delta measure effectively identifies layers where substantial parameter updates occur during fine-tuning. Therefore, we consider layers that contribute less to updating the class token as less critical, and we assign them lower compute budgets.

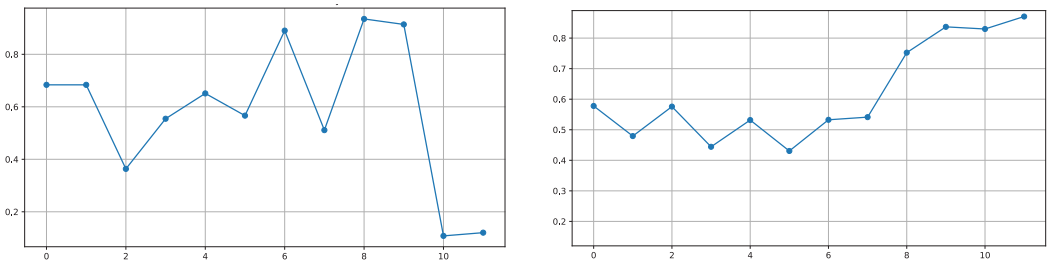


Figure 3.14. The Pearson's rank correlation between gradient norm and Δ_l^i for layers in DeiT-S (left) and ViT-B (right). x axis represents transformer layers.

Adaptive budget assignment: At fine-tuning step i, layer l updates the class token according to:

$$\text{CLS}_l^i = \mathcal{F}^l(\text{CLS}_{l-1}^i) \tag{3.16}$$

where CLS_{l-1}^i is the class token representation coming from the previous layer, $\mathcal{F}^l$ represents the layer l, and CLS_l^i is the updated class token representation. To capture the variations in the class token embedding, we define the class token delta at layer l and training step i as:

$$\Delta_l^i = ||\text{CLS}_l^i - \text{CLS}_{l-1}^i||_2^2 \tag{3.17}$$

The class token delta quantifies the magnitude of the update performed by layer l on the class token's residual stream. We use the variation in class token deltas across iterations to compute the training budget. At a training iteration i, we compute:

$$b_l^i = b_l^{i-1} + (\Delta_l^i - \Delta_l^{i-1}) \times \alpha \tag{3.18}$$

Here, b_l^{i-1} represents the budget from the previous training iteration, and α is the budget learning rate, which we initialize to match the fine-tuning learning rate. We always initialize the budget as 1 for each layer, and clip it during fine-tuning to have a value between 0 and 1. By assigning a budget to each layer, we obtain a distribution of budgets $\mathcal{D}_L^i$, where L is the number of layers and i is the current training step. We provide an example of how compute budgets vary during fine-tuning in Section 3.2.6.

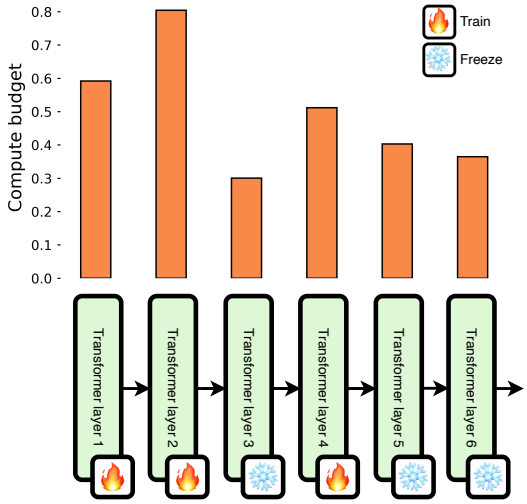


Figure 3.15. At each fine-tuning step, we assign what we call "compute budgets" to transformer layers. The budget determines the computational resources we invest in each layer, i.e., (a) whether the layer is frozen or trainable and (b) how many tokens that layer can process. By adaptively allocating the budget, we make the fine-tuning faster and more efficient in terms of FLOPs, memory, and time. In this example, we are training the top K layers, with K=3.

Freezeing a layer: To reduce memory usage and improve computational efficiency, we freeze the K lowest layers that are assigned the smallest budgets, where K is a hyperparameter. Freezing a layer means its parameters are no longer updated during training, which reduces the number of trainable parameters and eliminates the need to store its intermediate activations.

More precisely, given a budget distribution $\mathcal{D}_L^i$ over layers for a given iteration i, we sample K layers without replacement according to this distribution—ensuring each selected layer is unique. Only these K layers with the highest allocated budgets are kept trainable, while the remaining layers are frozen. We show an example of this procedure for $K = 3$ in Figure 3.15. Our probabilistic sampling approach for layer freezing offers several advantages over deterministic top-K selection. While deterministic selection (always choosing layers with highest budgets) might provide more stability, probabilistic sampling enables better exploration of the layer space and prevents premature convergence to suboptimal freezing patterns. This stochastic approach allows layers with moderate budgets to occasionally be selected, potentially discovering beneficial freezing configurations that deterministic methods might miss.

Selecting the Tokens

As pointed out in several works [276, 394, 66], not all tokens carry equally valuable information for the task at hand. To discard less useful tokens at each layer l, we leverage the budget distribution, computed as described in the previous section. Given an input sequence $T \in \mathbb{R}^{N \times E}$, where N is the sequence length and E is the embedding dimension, we only allow $b_l^i \cdot N$ tokens to flow to the next layer, where l is the transformer layer and i is the current fine-tuning iteration. Because b_l^i is always in $[0, 1]$, this is equivalent to selecting $(b_l^i \cdot 100)\%$ of the tokens from the input sequence.

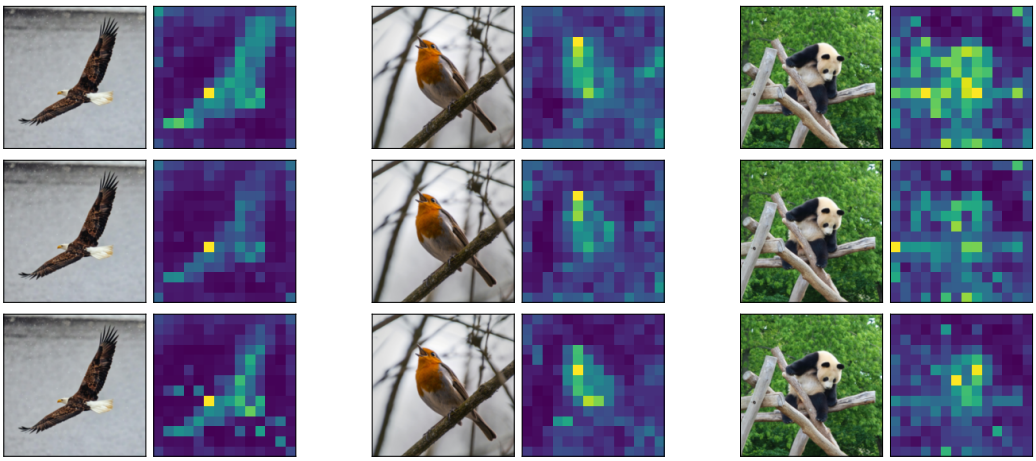


Figure 3.16. Attention of CLS token for different patches at layer 2,4,6 of DeiT-S [571]. Brighter patches have higher attention. CLS token's attention captures semantically important patches. We use the class token's attention as a measure of token importance to decide which patches to maintain in the residual stream.

To determine which tokens to discard, we follow [352], and rank the tokens based on their attention scores from the CLS token, retaining only the top $b_l^i \cdot N$ tokens. Tokens with higher attention scores are more influential in determining the class prediction, as we show in Figure 3.16, where we plot some representative attention maps with respect to the class token for DeiT-S. By retaining only the most important tokens - those that contribute significantly to the class token's latent representation - we reduce computational overhead and memory usage while preserving the most relevant information for classification.

Our token selection mechanism draws theoretical inspiration from mixture-of-experts (MoE) routing principles, where computational resources are dynamically allocated based on learned importance scores. In MoE architectures, a gating network determines which experts should process each token, typically using a top-k selection strategy to maintain sparsity while preserving model capacity [531, 182]. Just as MoE routing preserves model expressiveness by activating relevant experts, our token selection preserves sequence-level information by retaining high-attention tokens. The sparsity induced by top-k selection in both paradigms reduces computational overhead while maintaining performance, following the principle that most information is concentrated in a subset of available computational paths.

In preliminary tests, we let unattended tokens proceed to the next layer. However, analysis of attention score distributions revealed that tokens excluded at one layer remained excluded in subsequent layers. Consequently, we opted to discard the least attended tokens, preventing their progression to the next layer. This approach further reduced batch size and memory consumption without performance loss.

3.2.4 Validation of Token Selection approach

To validate the robustness of our CLS-centric token selection approach, we conducted an attention entropy analysis across all transformer layers. The entropy of attention

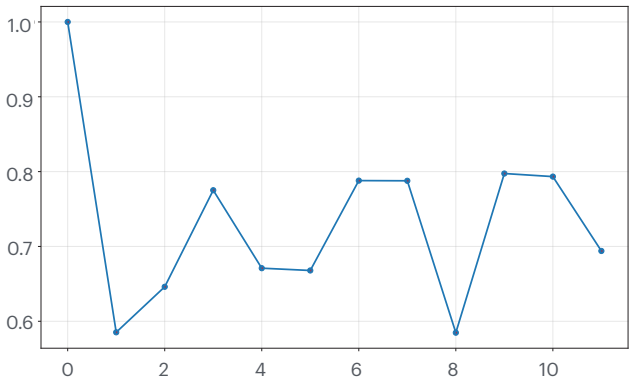


Figure 3.17. Average entropy of attention scores in DeiT-T, per each layer.

Model	embedding	#heads	#layers	#params
DeiT-T	192	3	12	5M
DeiT-S	384	6	12	22M
ViT-B	768	12	12	86M

Table 3.1. Variants of tested ViT architectures.

scores serves as a measure of attention diversity, where higher entropy indicates more distributed attention across tokens, while lower entropy suggests concentrated attention on specific tokens. Our analysis, as shown in Figure 3.17 reveals that from the second layer onward, the attention entropy remains relatively stable and maintains sufficient diversity, indicating that our token pruning strategy preserves meaningful attention patterns throughout most of the network depth. However, we observe a notable deviation in the first layer, where attention entropy exhibits different characteristics compared to subsequent layers. This finding aligns with prior observations that early transformer layers often exhibit distinct attention behaviors, potentially focusing on low-level features or exhibiting less task-specific attention patterns.

3.2.5 Experimental Setup

We fine-tune our models, pre-trained on Imagenet [144], on Flower-102 [429], Cifar-100 [310] and the more challenging Food-101 [70] dataset. We test a larger pre-trained Vision Transformer ViT-B [167] along with the DeiT-S and DeiT-T [571] models, that are smaller and more parameter efficient, thus more likely to be used in resource-constrained scenarios. We show details about the tested model architectures in Table 3.1. We download pre-trained weights from timm-models [608]. In all the runs, we set K (number of trainable layers) to 9, but we show results for other values in Section 3.2.6. We use mixed precision training [401] for all our experiments to keep memory usage as small as possible and simulate a real-world on-device training. During training, we keep track of FLOPs (Floating Point Operations), memory load and wall-clock time. While memory and time are important metrics to assess practical performance and resource utilization, they can be partially influenced by hardware. Therefore, we also use FLOPs , calculated over the whole architecture, to provide a hardware-independent

Table 3.2. Fine-tuning DeiT-T on Food-101 and Cifar-100. Best results in bold. Accuracies are shown with standard deviation across 3 seeds.

DeiT-T	Food-101				Cifar-100			
	C	M	T	Acc	C	M	T	Acc
FT head	**7.5**	**2389**	**13.44**	0.59±0.05	**5.0**	**2389**	**8.18**	0.66±0.06
FT full	11.2	3267	24.13	**0.83±0.03**	7.8	3267	14.14	**0.84±0.01**
FT top-3	9.6	2525	15.17	0.80±0.01	6.3	2525	9.13	0.82±0.09
ToME [66]	7.5	2983	18.06	0.82±0.11	7.1	2983	11.32	**0.84±0.09**
BSR [515]	8.5	2500	14.30	0.80±0.01	6.0	3052	8.45	0.82±0.02
LoRA [266]	8.2	2620	26.44	0.78±0.05	5.5	2643	15.10	0.81±0.02
ALaST (ours)	**5.1**	**1724**	**18.0**	**0.81±0.05**	**5.0**	**2397**	**11.32**	**0.83±0.07**

C=Compute (PFLOPs), **M**=Memory (MB), **T**=Time (minutes), **Acc**=Top-1 Accuracy

Table 3.3. Fine-tuning DeiT-S on Food-101 and Cifar-100. Best results in bold. Accuracies are shown with standard deviation across 3 seeds.

DeiT-S	Food-101				Cifar-100			
	C	M	T	Acc	C	M	T	Acc
FT head	10.5	4751	**18.36**	0.70±0.01	**18.4**	4751	**11.34**	0.75±0.04
FT full	30.2	7186	33.08	**0.86±0.04**	27.1	7186	20.14	**0.88±0.02**
FT top-3	22.8	5202	20.40	0.84±0.02	21.0	5202	12.51	0.87±0.08
ToME [66]	**10.0**	**2134**	25.05	0.80±0.12	26.7	**3122**	35.23	0.88±0.08
BSR [515]	12.3	3520	19.38	0.83±0.01	13.0	3500	12.13	0.87±0.02
LoRA [266]	22.5	5060	35.12	0.83±0.10	22.3	5060	32.00	0.85±0.09
ALaST (ours)	**14.5**	**4190**	**25.05**	**0.86±0.06**	**15.5**	**3613**	**15.05**	**0.87±0.11**

C=Compute (PFLOPs), **M**=Memory (MB), **T**=Time (minutes), **Acc**=Top-1 Accuracy

measure of computational complexity, enabling consistent cross-system comparisons. For reproducibility, we provide detailed descriptions of our experimental setup, including code, hyperparameters, and training procedures in 3.2.8.

3.2.6 Results

We evaluate ALaST across a diverse set of baselines, ranging from standard fine-tuning procedures to recent parameter-efficient techniques. Results are reported in Table 3.2, Table 3.3 and Table 3.4, with metrics including final accuracy, training FLOPs, peak memory usage, and wall-clock time.

We begin by comparing ALaST to two canonical fine-tuning strategies. In Full Fine-Tuning (FT Full), all model weights are updated, resulting in high accuracy at the cost of substantial compute and memory. In contrast, Head Fine-Tuning (FT Head) updates only the final classification layer, offering efficiency but typically much lower accuracy. As shown in Figure 3.19, ALaST offers a trade-off: it retains performance close to FT Full while using only **60%** of the FLOPs, **50%** of the memory, and **80%** of the training time, without adding any parameters to the model. We also benchmark against more recent and advanced methods. Token Merging (ToMe) [66] reduces memory by merging tokens according to a fixed schedule, but this static policy often limits performance. Block Selective Reprogramming (BSR) [515] freezes a preselected set of layers and prunes

Table 3.4. Fine-tuning ViT-B on Food-101 and Cifar-100. Best results in bold. Accuracies are shown with standard deviation across 3 seeds.

ViT-B	Food-101				Cifar-100			
	C	M	T	Acc	C	M	T	Acc
FT head	**80**	9657	**35.47**	0.83 ± 0.06	**70**	9657	**22.40**	0.84±0.08
FT full	150	13012	64.37	**0.90±0.06**	90	13012	40.34	0.90±0.12
FT top-3	120	7096	39.45	0.90±0.07	80	7096	24.11	**0.92±0.11**
ToME [66]	153	**4493**	48.42	0.88±0.07	75	**4493**	30.29	0.86±0.05
BSR [515]	117	6900	37.46	0.88±0.05	79	6900	23.30	0.91±0.04
LoRA [266]	109	11823	48.35	0.89±0.15	80	11823	32.10	0.90±0.12
ALaST (ours)	**107**	**5451**	**48.42**	**0.90±0.12**	**70**	**5341**	**29.30**	**0.90±0.13**

C=Compute (PFLOPs), **M**=Memory (MB), **T**=Time (minutes), **Acc**=Top-1 Accuracy

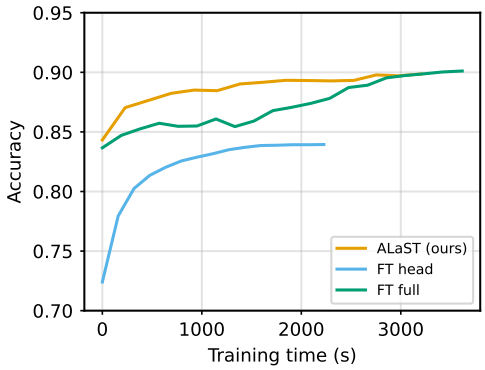


Figure 3.18. Comparison for ViT-B on a fixed data budget of 20 epochs for Food-101. Our method converges faster than the full fine-tuning with comparable accuracy.

tokens accordingly, requiring prior exploration to identify relevant blocks. Similarly, FT top-3 updates only the three most influential blocks, as identified through exploratory analysis. LoRA [266] sidesteps this manual selection by introducing trainable low-rank adapters, though this comes with additional parameters. In contrast, ALaST adaptively learns which layers to prioritize during training, avoiding the need for fixed schedules or block selection heuristics. Unlike LoRA, it introduces no additional trainable weights.

We find that LoRA performs inconsistently across model sizes: its accuracy drops on smaller architectures like DeiT-T and DeiT-S, likely due to limited representational capacity, which constrains the ability of additional parameters to adapt effectively to new data distributions. On larger models such as ViT-B, LoRA narrows the performance gap with ALaST. Since our method is complementary to parameter-efficient fine-tuning (PEFT) strategies like LoRA, we explore how the two approaches can be combined in the following section.

Integration with LoRA

As we showed in the previous section, LoRA performs rather poorly when applied to smaller models, while ALaST converges faster and to higher accuracy. On larger ViT-B, on the other hand, LoRA achieves similar accuracy to our method, at a higher memory

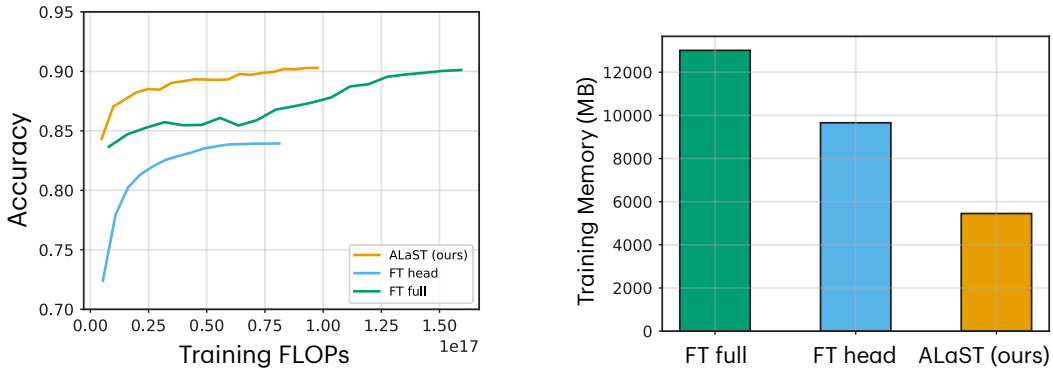


Figure 3.19. Comparison for ViT-B on a fixed data budget of 20 epochs for Food-101. Our method converges with fewer FLOPs (left), lower memory (right) with respect to fine-tuning all layers (FT full) and only the head (FT head).

Table 3.5. Combining LoRA + ALaST for fine-tuning ViT-B on Cifar-100 dataset.

Method	Compute (PFLOPs)	Memory (MB)	Time (min)	Accuracy
FT full	90	13012	40.34	0.90
LoRA	80	11823	32.10	0.90
ALaST	70	5341	29.30	0.90
LoRA + ALaST	**50**	**4243**	21.20	**0.89**

cost to store the activations for all layers. Because the two methods are orthogonal, we integrate them and test LoRA + ALaST on ViT-B.

In order to combine LoRA with ALaST, we apply the budget schedule to LoRA's additional parameters and keep the transformer layers frozen, except for the classfication head. We report results for Cifar-100 and Food-101 in Table 3.5 and Table 3.6, respectively. By intergrating LoRA with ALaST, we further reduce the memory footprint, while still achieving accuracy comparable to baseline.

We clarify that the "orthogonal optimization" in LoRA+ALaST refers to complementary optimization dimensions—parameter efficiency (LoRA) and computational efficiency (ALaST)—rather than implying additive accuracy improvements. As shown in Tables 5-6, the combination maintains competitive accuracy while achieving notable computational savings. For instance, ViT-B with LoRA+ALaST reduces memory usage from 5,451 MB to 4,243 MB compared to ALaST alone, demonstrating the primary benefit of reduced computational overhead rather than performance gains. This approach offers practitioners a method to achieve parameter efficiency without sacrificing the computational advantages of ALaST.

By integrating LoRA with ALaST, we achieve meaningful computational savings through reduced memory footprint (e.g., 4,243 MB vs. 5,451 MB for ViT-B) while maintaining accuracy comparable to baseline, demonstrating the practical value of combining orthogonal efficiency optimization approaches. The primary benefit of this combination lies in the computational savings achieved through reduced memory requirements while preserving model performance.

Table 3.6. Combining LoRA + ALaST for fine-tuning ViT-B on Food-101 dataset

Method	Compute (PFLOPs)	Memory (MB)	Time (min)	Accuracy
FT full	150	13012	64.37	**0.90**
LoRA	109	11823	48.35	0.89
ALaST	107	5451	48.42	0.90
LoRA + ALaST	**95**	**4243**	41.00	**0.90**

K	Compute (PFLOPs)	Memory (MB)	Time (min)	Accuracy
1	11.6	3125	20	0.81
2	11.9	3211	21.30	0.83
3	12.0	3297	22.20	0.84
4	12.4	3383	23.50	0.85
8	13.0	3888	23.10	0.86
10	15.5	4000	24.10	0.86
12	15.8	4137	25.10	0.86

Table 3.7. Different values for the number of trainable blocks at each iteration, K.

Layer Budgets across Fine-tuning

Budget assignment in ALaST is dynamic and happens at each fine-tuning iteration. We examine now how the budget assignement varies during training for different models and datasets. For DeiT-S and ViT-B, higher compute budgets are usually allocated to the first layers already in the initial part of the fine-tuning, as we show in Figure 3.20. Interestingly, for the first two layers, the method allocates high budgets across the entire fine-tuning, while for central layers the budget usually exhibits a higher variance. For DeiT-T, we observe a different pattern, where first and last layers are allocated higher budgets, while central ones are often neglected. In the smaller DeiT-T, on the other hand, the budget is allocated mainly to initial and last layers, and the distribution results more peaked.

Number of trainable blocks

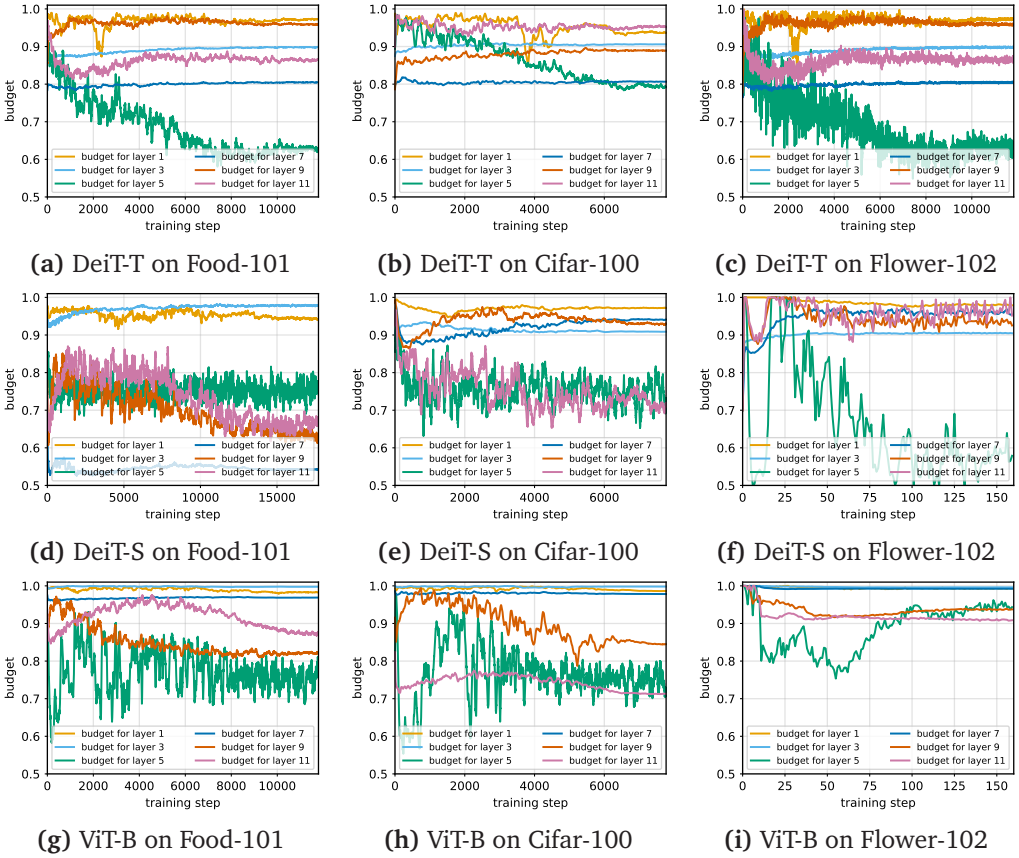
At each iteration, we select K transformer layers for fine-tuning. We showed experiments with $K = 8$. In Table 3.7, we report results for different values of K, showing the accuracy and FLOPs trade-off for each. We see that, in general, increasing the number of trainable layers leads to better performance. The improvement becomes smaller when K is greater than 8.

3.2.7 Additional Results

We show additional results on the Flower-102 dataset in Table 3.8.

3.2.8 Implementation details

In the following, we provide details on the implementation and fine-tuning hyper-parameters. We fine-tune the three models using Adam optimizer [302] with a batch



(a) DeiT-T on Food-101 **(b)** DeiT-T on Cifar-100 **(c)** DeiT-T on Flower-102

(d) DeiT-S on Food-101 **(e)** DeiT-S on Cifar-100 **(f)** DeiT-S on Flower-102

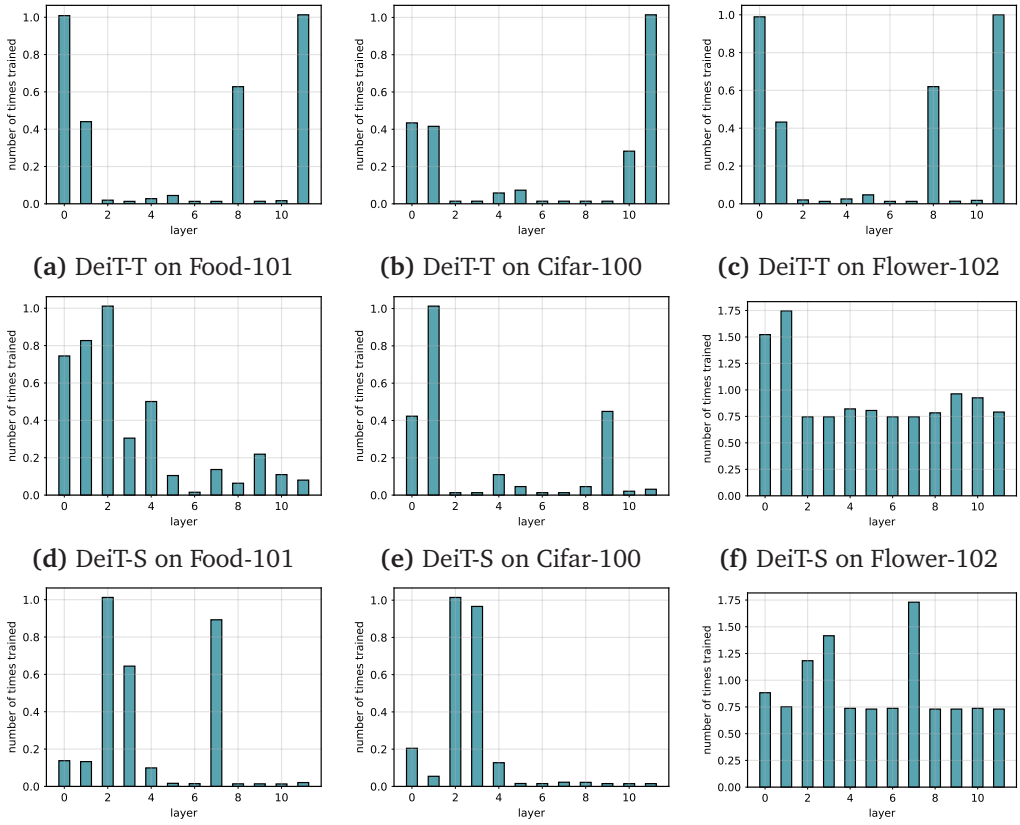
(g) ViT-B on Food-101 **(h)** ViT-B on Cifar-100 **(i)** ViT-B on Flower-102

Figure 3.20. Compute budget assignment during fine-tuning.

size of 128. After an initial grid search, we set the learning rate to 0.0001 for all methods but LoRA, where we found 0.001 to be more effective. We use two random augmentations picked between horizontal or vertical flipping and random cropping for all datasets and runs. All images are resized to 224×224, that is the required input size for the pretrained models. We download the weights of the models (pre-trained on ImageNet [509]) from Timm Image models [608]. During fine-tuning we keep track of FLOPs, peak memory and training time on an NVidia RTX4090 GPU. To measure FLOPs and memory load, we use PyTorch MACs counter and memory allocation tools [454]. Finally, we provide the Python code used for experiments.

3.2.9 Conclusion

We introduced ALaST, a simple and effective method to fine-tune ViTs in low-resource scenarios, saving computational budget, memory load and training time, with minimal modifications to the training pipeline. Although we test ALaST on ViTs, its principles are generalizable to other transformer-based architectures, which we plan to explore in future work.



(a) DeiT-T on Food-101 **(b)** DeiT-T on Cifar-100 **(c)** DeiT-T on Flower-102

(d) DeiT-S on Food-101 **(e)** DeiT-S on Cifar-100 **(f)** DeiT-S on Flower-102

(g) ViT-B on Food-101 **(h)** ViT-B on Cifar-100 **(i)** ViT-B on Flowes102

Figure 3.21. Final assignment on budgets for training on different datasets and models.

3.2.10 Limitations

Our experimental evaluation is limited to image classification tasks (CIFAR-100, Food-101, Flowers-102), which restricts the generalizability of our findings. Extension to dense prediction tasks such as object detection and semantic segmentation is challenging because our method relies on the CLS token for importance estimation. Dense prediction tasks require spatial reasoning across all patch tokens, making our current importance assessment strategy unsuitable without significant modifications. The approach also assumes that gradient-based metrics on the CLS token effectively capture layer importance, which may not hold across all architectures or tasks.

Table 3.8. Comparison of Fine-tuning DeiT-T DeiT-S and ViT-B on Flower-102 dataset.

Model	DeiT-T				DeiT-S				ViT-B			
	C	M	T	Accuracy	C	M	T	Accuracy	C	M	T	Accuracy
FT head	0.12	2389	**1.11**	0.51±0.11	**0.3**	4751	1.5	0.54±0.3	**1.0**	9657	**3.58**	0.73±0.1
FT full	0.14	3267	2.12	0.72±0.10	0.5	7186	2.4	**0.89**±0.4	1.5	13012	7.18	**0.97**±0.11
FT top-3	0.12	2525	1.19	0.55±0.05	0.4	5202	1.2	0.77±0.1	1.2	7096	4.25	0.90±0.08
ToME [66]	0.03	2983	1.39	**0.88**±0.3	0.5	2134	3.0	0.88±0.13	1.3	**4493**	5.31	0.95±0.08
BSR [515]	**0.11**	**2000**	1.15	0.73±0.3	0.39	**3520**	**1.1**	0.76±0.06	**1.0**	6900	4.12	0.90±0.18
LoRA [266]	**0.11**	2400	2.19	0.56±0.12	0.5	7256	2.5	0.71±0.13	**1.0**	11823	7.44	**0.97**±0.13
ALaST (ours)	**0.8**	**3820**	**1.39**	**0.79**±0.1	**0.3**	**3523**	**2.2**	**0.9**±0.06	**1.5**	**5341**	**5.31**	**0.97**±0.9

3.3 Adaptive Semantic Communication for Transformer Inference

Recently, there was a surge of interest in semantic and goal-oriented communications, establishing this paradigm as crucial for the development of 6G AI-native communication networks [549, 550]. In contrast to traditional communication systems, which focus on efficiently transmitting the data in the form of raw bits, semantic communication emphasizes understanding the meaning and intent behind the data, allowing for more efficient information exchange tailored to the specific goals of the communication task. This prioritizes the meaning and purpose behind transmitted data instead of the transmission itself and reduces the need for exact bit-level reconstructions, thereby offering gains in efficiency and adaptability, particularly in dynamic or resource constrained environments [510, 480, 59, 96, 162].

A prominent use case for semantic communication is edge inference, where distributed edge devices collect high dimensional sensor data and must convey only the most task relevant information to a nearby server for real time decision making [162]. In such scenarios, goal-oriented semantic communication enables these devices to transmit compressed, high level representations that are sufficient for the target inference task, such as object detection or anomaly recognition, while significantly reducing communication overhead. This is especially important in wireless edge networks, where constraints on bandwidth, latency, and power demand intelligent mechanisms to select and transmit only semantically meaningful features aligned with the task objective. In this context, transformer-based neural networks, with their inherent attention mechanisms and contextual processing capabilities, serve as ideal candidates for extracting and compressing semantic information from data into a lower-dimensional representation space [624]. This representation consists of multiple vectors called *tokens*, each capturing different semantic aspects of the input. By treating tokens as discrete transmission units, the system can selectively transmit only the most semantically relevant tokens while discarding less informative ones; this property has enabled the development of semantic token-based communication systems that adapt to dynamic channel conditions and application requirements, see, e.g. [154, 480, 479, 670].

Outline. This paper is structured as follows: Section II provides background on Vision Transformers (ViT), highlighting their token-based architecture and suitability for semantic communication. Section III introduces the system model, defining the communication framework between edge devices and servers. Section IV details the design of the proposed token-based DJSCC architecture, including adaptive token selection and compression mechanisms. Section V presents a dynamic optimization strategy based on Lyapunov stochastic control for real-time adaptation to varying channel conditions under bandwidth and noise constraints. Section VI reports empirical results that validate the performance and adaptability of the proposed system under different SNR regimes, offering qualitative insights into the semantic interpretability of token selection. Finally, Section VIII draws the conclusions and directions for future work.

Related work

Edge inference comes with the drawback of an ad-hoc deployment of computing resources, which might be more costly in terms of economic and environmental aspects,

if compared to a shared infrastructure. However, it allows end devices and service consumers to keep data local, thus promoting privacy and secrecy, and retrieving results within strict latency constraints, without the need of reaching distant central clouds. To achieve a positive balance between benefits and drawbacks, the joint optimization of communication and computing segments is a fundamental brick [162]. In general, edge inference can take place: i) locally at an end device, ii) fully offloaded to an edge server (or, Mobile Edge Host-MEH), or iii) partially performed at the two end sides [385]. The first solution (full local inference) typically requires model optimization techniques such as pruning or quantization [602], to be able to efficiently run a full (deep) model locally in a resource limited device. This comes with no communication overhead and full secrecy as no data is exposed, however at a cost of local computation and possibly reduced inference accuracy. The second solution (full offloading) [636, 98] relieves the device from any processing, but it can lead to high communication overhead and data exposure. In the third case (partial offloading), which is the most flexible one, as also highlighted in [328, 59], part of the computation is performed locally to extract relevant (possibly lower dimensional) features. This is helpful if the edge server is temporary unavailable, or to reduce the communication overhead needed to offload the inference task.

Several methodologies have been proposed to enable local feature extraction with the goal of minimizing communication overhead while maintaining acceptable task performance, typically measured in terms of accuracy. For a comprehensive overview, the reader is referred to the recent survey in [668]. One straightforward approach is to split a deep neural network (DNN) between a local device and an edge server, a technique commonly referred to as DNN splitting [385]. This splitting can be adapted dynamically based on channel conditions and the availability of radio and computational resources [328, 319]. From the perspective of task performance (e.g., achieving a target classification accuracy), prior work has shown that features extracted at different layers—corresponding to different splitting points—exhibit varying levels of robustness to channel noise and transmission errors [59]. This observation highlights the importance of selecting optimal split points depending on system constraints. When scaling to powerful architectures such as Vision Transformers, the opportunity for DNN splitting is further enriched by mechanisms like token pruning, which can significantly reduce communication costs by discarding less informative tokens. Moreover, when the model is trained end-to-end with the wireless channel modeled as a differentiable layer, DNN splitting effectively performs joint source and channel coding. This results in a learned *Deep Joint Source-Channel Coding* (DJSCC) scheme that simultaneously achieves compression and error resilience, optimized for the specific communication environment.

Recent advancements in wireless semantic communication have highlighted DJSCC as a promising approach, see, e.g., [632, 231, 642]. DJSCC systems integrate source and channel coding into a unified process over wireless channels, employing deep neural networks to tailor transmission strategies to specific application requirements. By modeling the communication channel as a differentiable layer within the network, these systems enable end-to-end learning that jointly adapts both source and channel coding, leading to improved performance across key metrics. DJSCC methods have shown clear advantages over traditional coding schemes, particularly in semantic communication settings, by exhibiting graceful performance degradation below information-theoretic

signal-to-noise ratio (SNR) thresholds and avoiding the sharp declines characteristic of conventional approaches [73, 371]. This makes DJSCC a compelling choice for practical scenarios characterized by low bandwidth and/or low SNR, where traditional methods struggle to maintain reliable performance.

Most existing JSCC systems operate under fixed trade-offs optimized for specific channel conditions and may struggle to generalize across different or fluctuating environments. However, in dynamic communication settings, where channel conditions can vary rapidly, the need for adaptive mechanisms that can adjust their operating parameters in real time becomes essential. To this end, recent studies have introduced adaptive methods that dynamically respond to real-time channel variations. These approaches include partitioning the model's latent space to support variable transmission rates [632], masking non-essential channels within convolutional architectures [644], and employing transformer-based architectures to project tokens onto variable-length representations [126]. Similarly, the model in [642] demonstrates superior coding efficiency compared to CNN-based JSCC and traditional BPG + LDPC schemes, while achieving lower end-to-end latency. Furthermore, [53] leverages transformer-based models in cooperative relay networks, showcasing notable beamforming gains. Additionally, several works have focused specifically on communication efficiency. For example, Wang et al. [595] developed an adaptive source allocation strategy based on quantization, enhancing transmission reliability under changing conditions. Similarly, Zhou et al. [685] introduced a Universal Transformer for semantic text transmission that adjusts its communication protocol depending on the channel environment. Finally, the work in [58] proposed a semantic-aware edge learning framework for multi-user wireless networks, where users adaptively decide between local and remote inference and transmit only task-relevant features via goal-oriented compression, while a Lyapunov-based optimization dynamically allocates communication and computational resources to balance energy, latency, and inference accuracy.

An additional degree of freedom in the design of DJSCC involves rendering the model adaptive, enabling it to dynamically adjust its computational complexity and output dimensionality, thereby enhancing flexibility and performance across diverse operating conditions [632]. Adaptive computation can be achieved through several approaches, broadly categorized into two strategies: i) employing multiple specialized models for different trade-offs [119, 614, 146, 6, 251]; or ii) embedding flexibility into a single model capable of handling a range of constraints. Here, we focus on the second scenario. Models in this category embed adaptivity into model architectures via dynamic networks that adjust processing based on input complexity [518]. Some methods employ sampling to process only relevant image regions within budget constraints [395, 239, 155], while others activate specific submodules conditionally [610]. Mixture-of-Experts approaches dynamically route inputs through specialized sub-networks [183, 532, 674], optimizing resource use. Early Exit techniques introduce intermediate classifiers to halt computation once sufficient confidence is achieved [470, 687, 239, 312].

Contributions of the Paper

In this work, we propose a novel transformer based DJSCC algorithm tailored for edge inference scenarios, where resource constrained devices must transmit task relevant information under dynamic channel conditions and limited computational budgets.

Building on the conditional computation framework introduced in [518] and extending our preliminary work in [154], we propose a DJSCC framework composed of multiple autoencoders with different compression factors, and a trainable *semantic token selection* mechanism that leverages the structural properties of transformer models to dynamically identify and transmit the most relevant input components based on relevance to the downstream task. Additionally, we propose a dynamic resource allocation algorithm based on stochastic optimization [424], which adaptively tunes the framework's parameters to maximize inference performance while accounting for communication overhead in time-varying conditions. Integrated with our DJSCC framework, this mechanism enables real-time adaptation of token selection and compression, improving the semantic efficiency of transmitted information in support of edge inference tasks.

The proposed framework offers three key advancements: (i) it compresses semantic tokens into a complex valued representation space while minimizing information loss during reconstruction at the receiver; (ii) it achieves robustness across a wide range of channel noise conditions using a single model, removing the need for multiple specialized DJSCC networks; and (iii) it operates in the complex domain while remaining compatible with standard real valued neural networks, allowing seamless integration with pre trained models. By reducing the number of tokens processed and transmitted, the semantic token selection mechanism also reduces the overall complexity of the overall DJSCC framework, improving its efficiency and scalability, particularly in resource constrained environments.

Through extensive comparative analysis, we demonstrate that our system achieves superior accuracy compression trade offs in edge inference settings by leveraging its additional degrees of freedom. The proposed token selection mechanism also facilitates the extraction of semantically rich and interpretable representations, laying the groundwork for interpretable by design models in next generation AI native communication systems. Its adaptability to varying channel conditions and task requirements makes it particularly well suited for edge devices, where efficient resource management is critical to achieving high performance inference.

3.3.1 Background on Vision Transformers

In this section, we provide a brief overview of the Vision Transformer (ViT) model [165], which serves as a core component of our proposed token-based communication framework, although the method can be readily adapted to scenarios involving other transformer-based architectures. We summarize recurring notation in Table 3.9.

A ViT model, denoted as $f(\mathbf{x})$, takes an image $\mathbf{x} \in \mathbb{R}^{C \cdot H \cdot W}$ as input, where C, H, and W represent the number of channels, height, and width of the image, respectively. The overall structure of the model is defined as follows:

$$f(\mathbf{x}) = \mathcal{C} \circ \mathcal{B}^L \circ \mathcal{B}^{L-1} \circ \cdots \circ \mathcal{B}^1 \circ \mathcal{E}(\mathbf{x}) \tag{3.19}$$

where $\mathcal{E}(\mathbf{x})$ is a preprocessing layer that turns the image into a sequence of tokens $\mathbf{H}^0 \in \mathbb{R}^{n \times d}$, with n the number of tokens and d their features; then, $\{\mathcal{B}^1 \dots \mathcal{B}^L\}$ are transformer blocks that process the tokens via multi-head attention (MHA) and feed-forward networks, with L denoting the number of blocks. The set of tokens is created by dividing an image into non-overlapping patches of fixed length, which are then flattened and projected to a fixed embedding size (i.e., d) using a trainable network. To preserve

Table 3.9. List of recurring symbols

Symbol	Description
$\mathbf{x}$	Input image of size $\mathbb{R}^p$ with $p = C \cdot H \cdot W$
$\mathcal{E}, \mathcal{B}, \mathcal{C}$	ViT blocks (resp. preprocessing, transformer block, classification head)
n	Number of initial tokens
α, n_α	User-defined budget, number of selected tokens
d	Original dimension of each token
o, r	Compressed dimension $o < d$, with ratio $r = \frac{o}{d}$
$\mathbf{H}^l$	Set of tokens at layer l of the ViT, of shape $n \cdot d$
$\mathbf{s}$	Symbols transmitted on the channel
$E(\mathbf{x})$, $D(\mathbf{s})$	DJSCC encoder and decoder
η	Channel (non-trainable)
$\mathcal{C}_E$, $\mathcal{C}_D$	Receiver encoder and decoder
$\boldsymbol{\gamma}(t)$	Optimized time-varying parameters $\{r(t), \alpha(t)\}$

spatial information, a unique learnable positional embedding is added to each token, encoding the original position of each patch within the image. Finally, the tokens set includes a trainable class token, which is used as input to the classification layer $\mathcal{C}$ to produce the final prediction vector.

The MHA mechanism represents the core of each transformer block. This mechanism consists of H parallel self-attention heads, each computing outputs by analyzing interactions between tokens in the input sequence. These outputs are then concatenated along the feature dimension and projected through a learnable matrix $\mathbf{W}_o \in \mathbb{R}^{Hd_v \times d}$, where d_v is the output dimension of each attention head and d is the model's hidden dimension. Letting $\mathbf{H}^{l-1} \in \mathbb{R}^{n \times d}$ denote the sequence of token embeddings input to the l-th transformer block, the MHA function reads as:

$$\mathrm{MHA}^l(\mathbf{H}^{l-1}) = \left[\mathrm{SA}_1(\mathbf{H}^{l-1}), \ldots, \mathrm{SA}_H(\mathbf{H}^{l-1})\right]\mathbf{W}_o \tag{3.20}$$

for $l = 1, \ldots, L$, where SA_i is the i-th Self-Attention head:

$$\mathrm{SA}_i(\mathbf{H}^{l-1}) = \mathrm{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^\top}{\sqrt{d_k}}\right)\mathbf{V}_i, \tag{3.21}$$

for $i = 1, \ldots, H$, where the softmax is applied row-wise. Each attention head in Eq. (3.21) operates using three learned projections: Query ($\mathbf{Q}_i$), Key ($\mathbf{K}_i$), and Value ($\mathbf{V}_i$), computed as follows:

$$\begin{aligned}
\mathbf{Q}_i &= \mathbf{H}^{l-1}\mathbf{W}_{q,i}, & \mathbf{W}_{q,i} &\in \mathbb{R}^{d \times d_k},\\
\mathbf{K}_i &= \mathbf{H}^{l-1}\mathbf{W}_{k,i}, & \mathbf{W}_{k,i} &\in \mathbb{R}^{d \times d_k},\\
\mathbf{V}_i &= \mathbf{H}^{l-1}\mathbf{W}_{v,i}, & \mathbf{W}_{v,i} &\in \mathbb{R}^{d \times d_v},
\end{aligned}$$

where d_k is the dimension of the queries and keys. The resulting matrices $\mathbf{Q}_i$, $\mathbf{K}_i$, and $\mathbf{V}_i$ have dimensions (n, d_k), (n, d_k), and (n, d_v) respectively. After computing all H attention outputs, they are concatenated to form a matrix of shape (n, Hd_v) in Eq. (3.20), which is then linearly projected back to dimension d using $\mathbf{W}_o$. A key observation is that while each self-attention output preserves the dimensions of its input, only the feature dimension d must remain fixed throughout the model. The token count n, however, can be dynamically adjusted without compromising the MHA mechanism.

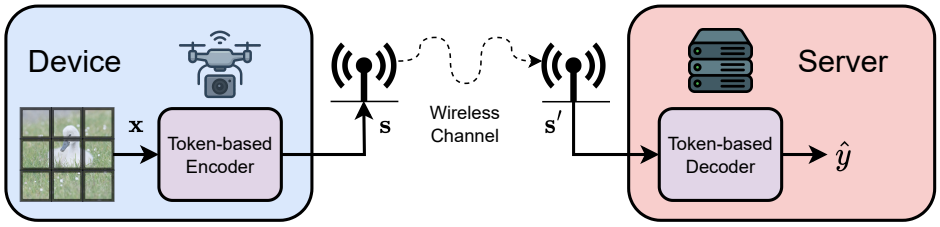


Figure 3.22. System scenario for semantic-oriented edge inference.

In the sequel, we exploit this flexibility to develop a token selection mechanism that identifies and preserves only the most semantically relevant tokens based on a user-defined computational budget and input characteristics. This approach offers three significant advantages. First, the computational burden is reduced, as the model processes a smaller number of tokens. Second, the selected tokens can be interpreted as the most meaningful part of the input image, thus providing useful insights into the model's internal reasoning. Last, in a semantic communication setting, transmitting only the most informative tokens enhances communication efficiency by reducing bandwidth requirements while preserving the core semantic content, enabling more reliable and meaningful exchanges between sender and receiver.

3.3.2 System Model

In this section, we present our adaptive token-based DJSCC system in three parts: the communication model (Section 3.3.2), the token compression architecture (Section 3.3.2), and the adaptive control mechanism (Section 3.3.2). The latter enables real-time optimization of transmission parameters based on channel conditions, as detailed in Section 3.3.4.

Communication model

We consider a communication scenario with a single edge device equipped with a camera that captures images represented by arrays $\mathbf{x} \in \mathbb{R}^p$, with $p = C \cdot H \cdot W$. The device applies a token-based DJSCC function $E : \mathbb{R}^p \to \mathbb{C}^q$ to produce the transmitted signal $\mathbf{s} = E(\mathbf{x}) \in \mathbb{C}^q$, subject to an average power constraint:

$$\frac{1}{q}\|\mathbf{s}\|_2^2 \leq 1, \tag{3.22}$$

where q denotes the available number of transmitted symbols. The compression ratio of the system is defined as:

$$\rho = \frac{q}{p}, \tag{3.23}$$

capturing the fraction of the input dimensionality that is retained in the transmitted representation. This allows the encoder to adapt the level of compression to meet resource constraints while preserving task-relevant semantics. Then, the signal $\mathbf{s}$ is transmitted over a wireless channel to a nearby edge server, which performs a downstream inference task such as classification or retrieval. At the server side, a token-based deep joint source channel decoder $D : \mathbb{C}^q \to \mathcal{Y}$ maps the received signal to an output $\hat{y} \in \mathcal{Y}$, where $\mathcal{Y}$

denotes the output space (e.g., class labels). Figure 3.22 provides an illustration of the proposed edge communication setting.

Communication occurs over a point-to-point complex additive white Gaussian noise (AWGN) channel. The received signal is given by

$$\mathbf{s}' = h\mathbf{s} + \mathbf{n}, \tag{3.24}$$

where $h \in \mathbb{C}$ is the channel gain and $\mathbf{n} \in \mathbb{C}^q$ is an additive noise vector with entries drawn i.i.d. from a complex normal distribution $\mathcal{CN}(0, \sigma_h^2)$. For the AWGN channel, we set $h = 1$. The signal-to-noise ratio (SNR) for the communication of vector $\mathbf{s}$ can then be expressed as $\text{SNR} = |h|^2\|s\|^2/\sigma_n^2$.

Token-based DJSCC Design

In this section, we introduce our token-based DJSCC architecture, which incorporates an adaptive token selection and compression mechanism. This design enables dynamic control over output resolution, computational cost, and communication load by selectively retaining and transmitting only the most informative tokens. Specifically, we consider a DJSCC scenario where the entire communication pipeline is described by an end to end deep neural network composed of multiple macro-blocks:

$$f^{\text{jscc}}(\mathbf{x}) = D \circ \eta \circ E(\mathbf{x}), \tag{3.25}$$

where again $\mathbf{x} \in \mathbb{R}^{C \cdot H \cdot W}$ is the input image, while E and D are, respectively, the transmission and the receiver pipelines, and η the channel in Eq. (3.24), modeled as a non trainable layer. Both the transmitter and receiver are constructed based on a pre-trained ViT. Given a ViT with L transformer blocks, as defined in Eq. (3.19), we partition the model by selecting a splitting point $1 < s < L$. The first s blocks are assigned to the encoder E, while the remaining $L - s$ blocks form the decoder D. Then, the encoder E is augmented by with a novel component denoted $\mathcal{C}_E$. It is added at the end of the encoder to compress the features of the output tokens. Specifically, it consists of two components, c_R and c_I, which generate the real part and the imaginary part of the complex-valued symbols, respectively. The resulting symbols are then normalized to satisfy the power constraint specified in Eq. (3.22). Mathematically, we have:

$$E = \mathcal{C}_E \circ \left(\mathcal{B}^s \circ \cdots \circ \mathcal{B}^2 \circ \mathcal{B}^1\right) \circ \mathcal{E}. \tag{3.26}$$

where we show in red the added components. At the receiver side, the representation is first processed by a compression decoder $\mathcal{C}_D$, which transforms the received complex symbols back into real-valued features. This is achieved by extracting the real and imaginary components of the input, concatenating them, and processing the result with a lightweight neural network c_D. Then, the result is passed through the remaining transformer blocks, i.e.,

$$D = \mathcal{C} \circ \mathcal{B}^L \circ \cdots \circ \mathcal{B}^{s+1} \circ \mathcal{C}_D. \tag{3.27}$$

where again we highlight in red the components which are added to the pre-trained ViT model.

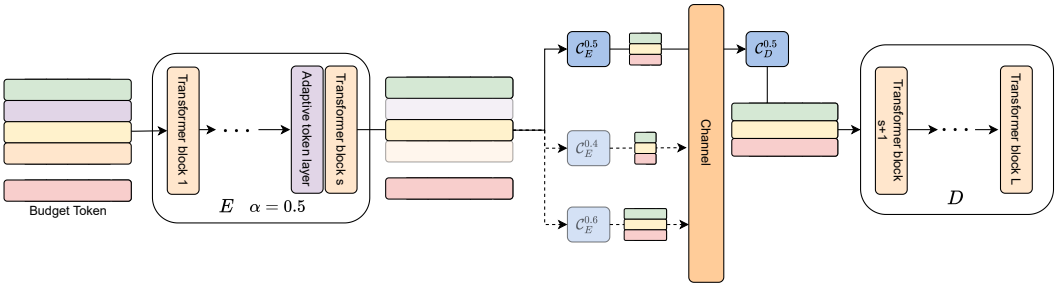


Figure 3.23. Schema of the proposed token-based DJSCC.

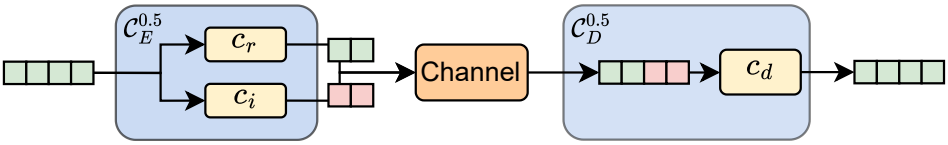


Figure 3.24. Overview of the proposed token-based transmission pipeline. The features of a generic token are processed by the compression encoder $\mathcal{C}_E^r$, having a compression ratio of $r = 0.5$, to produce the complex symbols, which are transmitted through a channel. The compression decoder $\mathcal{C}_D^r$ receives the symbols, concatenates imaginary and real parts, and recreates the features.

Adaptive DJSCC design

In a token-based communication system, the output of the encoder $E(x)$ determines the amount of data to be transmitted on the channel, which, in the most basic form, is given by the floating point numbers that must be encoded. In this paper, we propose exploiting the flexibility of transformer networks to design a model that can adapt both n and d in the output, thereby allowing tailoring them to the channel conditions. This is done by conditioning the encoder with a set of user-given parameters γ, such that its output depends also on it as $E(x, \gamma)$.

The last stage of the encoder E in Eq. (3.26) applies a token compression mechanism, denoted by $\mathcal{C}_E$. Instead of a single mechanism, we can adopt an ensemble of them, each one with a different compression factor r, each of which is defined as $\mathcal{C}_E^r : \mathbb{R}^{n \times d} \to \mathbb{C}^{n \times o_r}$, having a compression ratio $r = \frac{o_r}{d} \in (0, 1]$, and $o_r \leq d$; on the receiver side, we have the counterpart $\mathcal{C}_D^r$. Formally, the output of the encoder (after vectorization) yields the transmitted symbol vector which depends on r, as $\mathbf{s} \in \mathbb{C}^q$, with $q = n \cdot o_r$. A DJSCC trained with multiple tuples $(\mathcal{C}_E^r, \mathcal{C}_D^r)$, each with a different $r \in (0, 1]$, is adaptable and its parameter set is $\gamma = \{r\}$. Selecting the desired compression ratio is the first degree of adaptability and operates on the size of each token. Figure 3.24 shows the components of a mechanism having a compression ratio of 0.5.

The second degree of freedom exploits a key property of ViTs: the ability to discard or merge tokens during the forward pass, resulting in fewer output tokens while maintaining without detriment the ability to solve a task. To this end, we add a dynamic block $\mathcal{S}$ after each layer of the ViT, which removes some tokens from one layer to another. The Eq. (3.26) becomes:

$$E = \mathcal{C}_E \circ \left(\mathcal{S}^s \circ \mathcal{B}^s \circ \cdots \circ \mathcal{B}^2 \circ \mathcal{S}^1 \circ \mathcal{B}^1 \right) \circ \mathcal{E}. \tag{3.28}$$

This not only reduces the number of output tokens, but often also the computational cost of running the model itself. To control this behavior, we add a parameter α regulating the number of tokens produced by a generic dynamic ViT, so that in this case the set of parameters is $\gamma = \{\alpha, r\}$. In this case, The encoder also integrates the adaptability of the tokens' size, and its compression mechanism becomes $\mathcal{C}_E^r : \mathbb{R}^{n_\alpha \times d} \to \mathbb{C}^{n_\alpha \times o_r}$, where $n_\alpha = \mathcal{N}(\mathbf{x}, \alpha)$ is the number of tokens outputted by the adaptive ViT, for a generic input x.

Importantly, we fine-tune a model to work equally well for any configuration, giving us a unique model able to output a (variable) number of tokens of (variable) dimensionality. Once a model is fine-tuned, we design a dynamic optimization problem (Section 3.3.4) to map in an efficient way from the sensed channel conditions at time t (e.g., SNR) to an optimal configuration $\gamma(t)$ for the current time step t, to ensure at the same time that two properties are satisfied over a long-term horizon: (a) high accuracy of the model, and (b) optimal compression rate w.r.t. the channel conditions (corresponding to a choice of configuration giving rise to the smallest possible output on average).

An overview of the proposed dynamic token-based DJSCC system is depicted in Figure 3.23.

3.3.3 Adaptive Semantic Token Selection

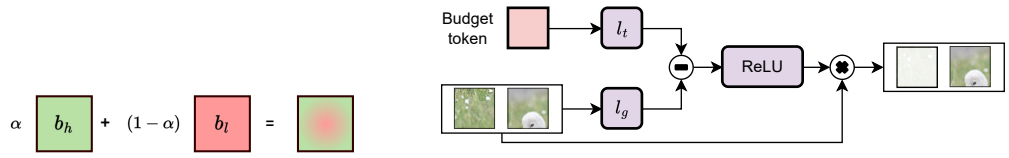
We begin our discussion with the token selection module $\mathcal{S}$ in Eq. (3.28), which extends the ViT formulation in Eq. (3.19) by dynamically retaining a subset of tokens at each transformer block $l = 1, \ldots, s$. This selection is governed by a computational budget constraint, specified by a scalar $\alpha \in (0, 1]$, which denotes the proportion of tokens to be retained. In other words, α controls the percentage of total tokens the model preserves for a given input sample $\mathbf{x}$. Furthermore, we make explicit the dependence on the trainable parameter vector $\mathbf{w}$ and the budget constraint α in equation Eq. (3.25) by writing the function as $f_{\mathbf{w}}^{\text{jscc}}(\mathbf{x}, \alpha)$. Given labeled data pairs $(\mathbf{x}, y)$, the training problem can then be formulated as:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \mathbb{E}_{(\mathbf{x},y)} \mathcal{L}(f_{\mathbf{w}}^{\text{jscc}}(\mathbf{x}, \alpha), y) \\ \text{s.t.} \quad & |\mathcal{N}_{\mathbf{w}}(\mathbf{x}, \alpha) - \alpha \cdot n| \leq \epsilon, \quad \forall\, \alpha \in (0, 1], \end{aligned} \tag{3.29}$$

where n denotes the total number of tokens; $\mathcal{N}_{\mathbf{w}}(\mathbf{x}, \alpha)$ represents the number of output tokens produced by the model E for a given set of weights $\mathbf{w}$, budget α, and input sample $\mathbf{x}$; and finally, $\mathcal{L}(\hat{y}, y)$ represents the classification loss, e.g., cross-entropy. From now on, we omit $\mathbf{w}$ for the sake of clarity. Intuitively, the constraint in Eq. (3.29) ensures that the number of output tokens remains close to the target proportion $\alpha \cdot n$, within a small error ϵ.

To solve the constrained problem Eq. (3.29), we need to design a trainable token selection method that can be incorporated into the ViT model in Eq. (3.19). To this aim, we introduce a new trainable *budget token* that is appended to the input sequence of image tokens. Specifically, the budget token is formed by combining two trainable tokens: $\mathbf{b}_l \in \mathbb{R}^d$, representing a low-budget configuration, and $\mathbf{b}_h \in \mathbb{R}^d$, representing a high-budget configuration. Then, given a target α value, the budget token is a row-vector computed as:

$$\mathbf{b}^0 = \alpha \mathbf{b}_h + (1 - \alpha)\mathbf{b}_l.$$



(a) The budget computed as the combination of two trainable tokens corresponding to low (b_l) and high (b_h) budget. The value α is defined by the user.

(b) The proposed selection mechanism is applied before each transformer block. The budget token is used to calculate the threshold value, which is evaluated against each gate value to determine if a token must be discarded.

Figure 3.25. The details of two components of our proposed token discarding approach. On the left, the creation of the budget token, while on the right, how the tokens are actively discarded. More detail in Section 3.3.2.

When $\alpha \approx 0$, the budget token $\mathbf{b}^0$ approximates $\mathbf{b}_l$, encouraging the model to operate under the lowest possible budget. Conversely, when $\alpha = 1$, the model adopts the high-budget configuration $\mathbf{b}_h$. This behavior is illustrated in Figure 3.25a. This additional token is introduced by the preprocessing layer in Eq. (3.19) as follows:

$$\mathcal{E}(\mathbf{x}) = \begin{bmatrix} \mathbf{H}^0 \\ \mathbf{b}^0 \end{bmatrix} \in \mathbb{R}^{(n+1)\times d}$$

where $\mathbf{H}^0$ represents the initial sequence of tokens. The budget token $\mathbf{b}^0$ is appended to the end of this sequence and propagated through all blocks alongside the other tokens.

For each transformer block, we now introduce a selection mechanism that uses the budget token representation $\mathbf{b}^l$ at the l-th layer, along with the current set of tokens, to identify a subset of tokens to discard. A transformer block equipped with this token selection mechanism is referred to as *adaptive*. To this end, we introduce two trainable linear models followed by a sigmoid activation function, which work together to selectively mask out less important tokens: a threshold model l_t and a token gating model l_g. The threshold model $l_t(\cdot) \in [0,1]$ takes the current budget token $\mathbf{b}^l$ as input and outputs a scalar threshold. The token gating model $l_g(\cdot) \in [0,1]^{(n-1)}$ produces a gating score for each token embedding $\mathbf{h}_j^l$, where $\mathbf{h}_j^l$ denotes the j-th row of $\mathbf{H}^l$ for $j = 1, \ldots, n-1$, excluding the budget and class tokens, which are never discarded. These outputs are used to compute a differentiable mask over the input tokens. For each token embedding $\mathbf{h}_j^l \in \mathbf{H}^l$, the corresponding mask value M_j^l indicates whether the token is retained ($M_j^l > 0$) or masked out ($M_j^l = 0$). Specifically, for a generic token selection module $\mathcal{S}^l$, and for each token embedding $\mathbf{h}_j^l \in \mathbf{H}^l$, the corresponding mask value is computed as:

$$M_j^l = \text{ReLU}\left(l_g(\mathbf{h}_j^{l-1}) - l_t(\mathbf{b}^{l-1})\right) \cdot M_j^{l-1}, \tag{3.30}$$

for $j = 1, \ldots, n-1$, where $M_j^1 = 1$ for all j, indicating that all tokens are initially retained. For computational consistency, the mask values for the budget and class tokens are fixed to 1 at all layers. In Eq. (3.30), by recursively multiplying the current mask with the one from the previous layer, any token that is discarded remains inactive for the remainder of the network, forcing consistency. The masking process can be applied either before or after each transformer block; in what follows, we assume the former. A

Figure 3.26. As the images flow through the model (from left to right), some tokens, highlighted in red, are discarded.

Algorithm 1 Implementation of a Token selection module $\mathcal{S}$

Require: Set of token embeddings $\mathbf{H}$, budget token $\mathbf{b}$, previous mask M (default: $M = 1$), threshold model $l_t(\cdot)$, gate model $l_g(\cdot)$ (we omit all layer indices for brevity).

threshold $\leftarrow l_t(\mathbf{b})$ ▷ Scalar threshold
gates $\leftarrow l_g(\mathbf{H})$ ▷ One gate per token
$m \leftarrow \text{ReLU}(\text{gates} - \text{threshold})$
$m \leftarrow m \times M$ ▷ Apply previous mask recursively
return $\mathbf{H} \cdot m,\ m$ ▷ Mask the tokens

visual overview of the proposed token selection mechanism is provided in Figure 3.25b, highlighting how the model actively removes tokens at each stage. The corresponding pseudocode for the masking procedure is presented in Algorithm 1. Finally, Figure 3.26 illustrates an example of tokens being progressively discarded across consecutive blocks in the proposed adaptive token architecture, demonstrating its ability to retain only the semantically relevant information from the input image.

In the following section, we describe how the budget allocation process is trained to encourage the model to produce a number of tokens that closely matches the target budget α.

Design of Token Selection Penalties and Training

To enforce the computational budget constraints defined in Problem (3.29), we introduce two complementary regularization terms. First, given a token selection module $\mathcal{S}^l$, with $l \leq s$, we approximate the number of selected tokens by the average mask value, denoted by $\overline{M}^l$, and defined as:

$$\overline{M}^l = \frac{1}{n+1} \sum_{j=1}^{n+1} M_j^l.$$

where M_j^l is calculated following Equation (3.30). Then, we quantify the deviation from the target budget at block l using the penalty function:

$$B(l) = \text{ReLU}\left(\left| \overline{M}^l - \alpha \right| - \epsilon \right), \tag{3.31}$$

where α is the target budget and ϵ is a margin parameter that allows tolerance in meeting the budget constraint. The ReLU function in Eq. (3.31) ensures that penalties are incurred only when the deviation exceeds this margin. Then, from Eq. (3.31), we impose the computational budget using two complementary losses. We define two regularization terms to guide the budget-aware training process: (i) the budget loss

$B(s)$ in Eq. (3.31), applied to the final adaptive block s, which enforces that the number of tokens transmitted over the channel matches the target budget; and (ii) a sparsity-inducing regularization R, which encourages early token elimination and is defined as:

$$R = \frac{1}{s}\sum_{l=2}^{s} B(l). \tag{3.32}$$

The penalty term in Eq. (3.32) promotes computational efficiency by encouraging the model to discard irrelevant tokens as early as possible in the network. Crucially, our recursive mask construction ensures that once a token is discarded, it remains inactive in all subsequent blocks, simplifying the optimization and enhancing training stability.

By incorporating the penalties in Eq. (3.31) and Eq. (3.32) and recasting the constrained problem Eq. (3.29) using the Lagrangian approach, the training objective of the proposed model reads as:

$$\min_{\mathbf{w}} \ \mathbb{E}_{(\mathbf{x},y)}\, \mathcal{L}(f^{jscc}(\mathbf{x},\alpha), y) + \lambda_s B(s) + \lambda_r R \tag{3.33}$$

where λ_s and $\lambda_r \geq 0$ are hyperparameters to balance task performance, budget adherence, and computational efficiency. Additionally, to allow for a correct training process, we initialize l_g and l_t so that, at the beginning of the training, their outputs are, respectively, around 1 and 0 for each input. This guarantees that at the beginning of the training no tokens are discarded, avoiding initialization biases. During training, we enable the model to handle varying computational budgets by sampling the budget parameter α uniformly from the interval $(0, 1]$ for each training instance. This approach effectively trains the model across the full spectrum of budget constraints, allowing it to adapt to any specified budget given by the user at inference time. At inference, we discretize the learned masks by removing tokens with zero-valued mask entries, thereby achieving actual computational savings.

3.3.4 Dynamic Optimization of Token Selection and Compression

In the previous section, we presented the proposed token-based DJSCC mechanism, which provides design flexibility through two key parameters: the token budget α and the compression ratio r. In this section, we introduce a method for dynamically selecting the optimal set of parameters γ for DJSCC, as defined in Section 3.3.2. We introduce the optimization in the context of time-varying scenarios, where channel conditions evolve due to factors such as fading, blockages, and user mobility. The method relies on a long-term problem formulation that is boiled down to a sequence of deterministic problems based on instantaneous observation of context parameters (e.g., channels) and opportunistically defined state variables.

We consider a time-slotted scenario where one image $\mathbf{x} \in \mathbb{R}^p$ is generated at each time instant t. Each image is first pre-processed locally using the token-based DJSCC, and the resulting intermediate representation is then transmitted wirelessly. Finally, the images are classified remotely using a decoder model. The wireless channel varies over time, and is assumed to be constant during a whole time slot (i.e., block fading). This setting requires a dynamic connect-compute DJSCC protocol that can adapt to changing conditions while maintaining stable performance. The adapted parameters are the compression ratio r and the token budget α (if present), both of which are

related to the ratio of radio resources allocated to the user. The objective is to maximize classification accuracy under bandwidth-related constraints. To this aim, we define the following parameters: (i) the vector $\boldsymbol{\gamma}(t) = \{\alpha(t), r(t)\}$, which is subject to optimization and consists of the compression ratio $r(t) \in (0,1]$ and the allocated budget $\alpha(t) \in (0,1]$; (ii) the compression factor $\rho(\boldsymbol{\gamma}(t)) = \frac{q(t)}{p}$, where $q(t)$ denotes the number of symbols produced by the encoder $E(x, \gamma(t))$; and (iii) $\mathrm{SNR}(t)$, the instantaneous signal-to-noise ratio at time slot t. We further define: (iv) $\Lambda(\boldsymbol{\gamma}(t), \mathrm{SNR}(t))$, a proxy function representing the model's accuracy as a function of the parameter configuration $\boldsymbol{\gamma}(t) \in \mathcal{C}$, and depending on the instantaneous channel condition via $\mathrm{SNR}(t)$; and (v) $\Gamma_{\mathrm{th}} \in (0,1]$, a constraint on the compression factor, representing the maximum allowed average ratio of transmitted symbols relative to the total input size, typically determined by bandwidth limitations.

The core objective of the algorithm is to dynamically select $\boldsymbol{\gamma}(t)$, i.e., the compression ratio $r(t)$ and the budget $\alpha(t)$, in order to maximize inference accuracy while satisfying a long-term constraint on the average number of transmitted symbols. Mathematically, this is equivalent to solving the long-term problem formulation:

$$\begin{aligned} \max_{\boldsymbol{\gamma}(t)\in\mathcal{C}} \quad & \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[\Lambda(\boldsymbol{\gamma}(t), \mathrm{SNR}(t))] \\ \text{subject to} \quad & \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[\rho(\boldsymbol{\gamma}(t))] \le \Gamma_{\mathrm{th}} \end{aligned} \tag{3.34}$$

where the set $\mathcal{C}$ denotes the discrete set of feasible values resulting from all possible combinations of $r(t)$ and $\alpha(t)$. The expectation is taken with respect to the channel realizations, whose statistics are assumed to be unknown a priori. To handle the long-term expectation constraint, we resort to Lyapunov stochastic optimization [425]. We introduce a *virtual queue* $Z(t)$, which evolves according to the update rule:

$$Z(t+1) = \max\big(0,\, Z(t) + \mu(\rho(\boldsymbol{\gamma}(t)) - \Gamma_{\mathrm{th}})\big), \tag{3.35}$$

where $Z(0) = 0$, and $\mu > 0$ is a tunable step-size hyperparameter that controls the sensitivity of the queue dynamics. Then, the long-term constraint can be equivalently reformulated in terms of the *mean rate stability* of the virtual queue $Z(t)$, i.e., $\lim_{T\to\infty} \frac{\mathbb{E}\{Z(T)\}}{T} = 0$. This stability condition ensures that the original long-term constraint is satisfied. By translating the constraint into a stability condition, the problem becomes more tractable. To proceed, we define a Lyapunov function $\mathcal{L}(t) = \frac{1}{2}Z^2(t)$, and introduce the *drift-plus-penalty* function:

$$\Delta_p(t \mid Z(t)) = \mathcal{L}(t+1) - \mathcal{L}(t) - V \cdot \Lambda(\boldsymbol{\gamma}(t), \mathrm{SNR}(t)), \tag{3.36}$$

where $V > 0$ is a tunable parameter that balances the trade-off between optimizing performance and enforcing the constraint in (3.34). As shown in [425], minimizing a suitable upper bound of (3.36) ensures the boundedness of the Lyapunov drift, which in turn guarantees the mean rate stability of $Z(t)$, thereby satisfying the long-term constraint in Eq. (3.34). Based on this principle, we adopt a greedy slot-wise optimization strategy that minimizes the upper bound of (3.36) at each time slot. Following the derivations in [425] and [396], this approach leads to solving the following per-slot optimization

Algorithm 2 Dynamic Token Selection and Compression

Require: A proxy function $\Lambda(\boldsymbol{\gamma}, \text{SNR})$ for the accuracy, a constraint Γ_{th}, the hyperparameters V and μ, the feasible set $\mathcal{C}$ for the optimization parameters.

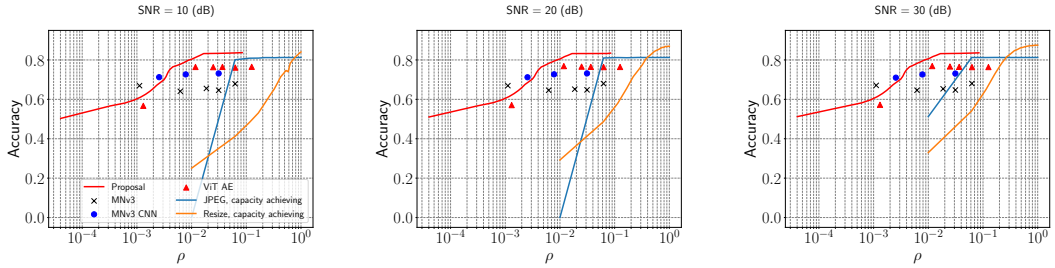
$Z(0) \leftarrow 0$
for $t \geq 0$ **do**
  Observe $\text{SNR}(t)$
  $\gamma^*(t) \leftarrow \underset{\boldsymbol{\gamma}(t)\in\mathcal{C}}{\arg\min} \; -V \cdot \Lambda(\boldsymbol{\gamma}(t), \text{SNR}(t)) + Z(t)\rho(\boldsymbol{\gamma}(t))$
  $Z(t+1) \leftarrow \max(0, Z(t) + \mu(\rho(\boldsymbol{\gamma}(t)) - \Gamma_{\text{th}}))$
end for

problem, which depends only on the current channel observation and the virtual queue state:

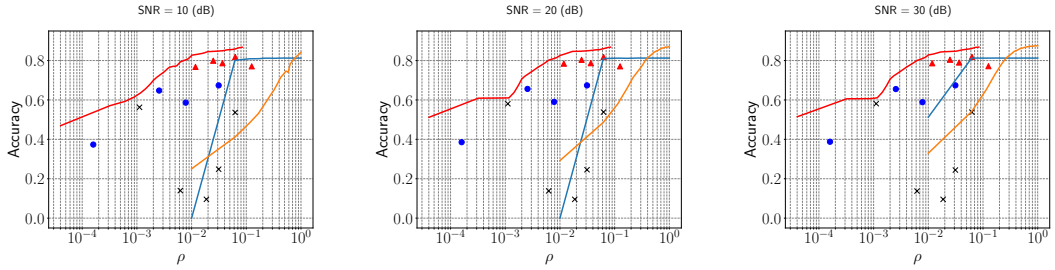
$$\min_{\boldsymbol{\gamma}(t)\in\mathcal{C}} \; -V \cdot \Lambda(\boldsymbol{\gamma}(t), \text{SNR}(t)) + Z(t)\rho(\boldsymbol{\gamma}(t)). \tag{3.37}$$

It is significantly simpler than the original formulation in Eq. (3.34), as it is deterministic at each time slot t and does not require prior statistical knowledge of contextual parameters (e.g., radio channel conditions). To solve this problem, we perform an exhaustive search over the set $\mathcal{C}$, which typically has low cardinality, to identify the optimal pair of parameters $\boldsymbol{\gamma}^*(t) = \{r^*(t), \alpha^*(t)\}$. Once the optimal solution is obtained, the virtual queue in Eq. (3.35) is updated using the instantaneous compression ratio $\rho(\boldsymbol{\gamma}^*(t)) = \frac{q^*(t)}{p}$, where $p^*(t)$ denotes the symbols produced by the optimal configuration of the encoder $E(x, \gamma^*(t))$. The pseudo-code that summarizes the proposed procedure is reported in Algorithm 2.

Accuracy's proxy function and Hyperparameter selection. To optimize the DJSCC system using Algorithm 2, we begin by constructing the proxy function $\Lambda(\boldsymbol{\gamma}, \text{SNR})$, which estimates the expected classification accuracy as a function of transmission parameters. Specifically, we systematically evaluate all combinations of compression ratio r, computational budget α, and a discretized set of SNR values over the training dataset. For each configuration, we compute the average classification accuracy, thereby creating a mapping $\Lambda\colon (\boldsymbol{\gamma}, \text{SNR}) \mapsto \text{Accuracy}$. This proxy enables efficient parameter selection during optimization by approximating model performance without requiring full inference. Following the construction of Λ, we perform hyperparameter tuning by evaluating multiple combinations of the control parameters V and μ via simulation. For each candidate pair, we execute the optimization procedure described in Algorithm 2 over $T = 10^5$ simulation steps. We then compute the average compression ratio $\Gamma(t)$ achieved over the final 10^3 steps of each run. The optimal hyperparameter configuration is selected as the one that maximizes the proxy accuracy Λ, subject to the compression constraint. These optimized hyperparameters are then deployed in operational settings, where each input sample arrives at a specific time slot t, and the channel conditions (SNR) may vary dynamically.



(a) Accuracy curves when using robust DJSCC models.



(b) Accuracy curves when using noiseless DJSCC models.

Figure 3.27. Classification accuracy versus compression ratio ρ at different (high) SNR levels, comparing the proposed method with the baselines under two training settings: (a) robust training with noisy channels; and (b) training with ideal (noiseless) channels. The legend, shown in the upper-left plot, is shared across all subfigures.

3.3.5 Empirical evaluation

In this section, we evaluate our proposed DJSCC framework across multiple experimental settings[4]. First, we describe our training methodology, detailing the model training, the integration of the communication pipeline, and the two distinct training scenarios: robust (with variable SNR) and noiseless. Next, we introduce our comprehensive set of baselines, including both neural network approaches and digital capacity-achieving methods. Finally, we present our experimental results and provide a detailed analysis of model performance across different channel conditions and compression settings.

Models pre-training. Each model is pre-trained and fine-tuned on the Imagenette dataset [262], which comprises 10,000 images divided into 10 classes. We evaluate three different architectures as backbones: a standard ViT [168], a MobileNetV3 [261], and our proposed Adaptive Semantic Token Selection approach. All models are trained for 150 epochs with a batch size of 256, using the Adam optimizer [303]. During training, we apply data augmentation techniques including RandAugment [121], Color Jitter, and random horizontal flipping with probability 0.5. For adaptive semantic token selection, we configure the first three blocks (i.e., $s = 3$) to be adaptive while keeping the rest of the model static. We set the hyperparameters to $\lambda_s = 2$ and $\lambda_r = 1$ in Eq. (3.33), and employ the budget sampling strategy detailed in Section 3.3.2. This configuration provides an optimal trade-off between token reduction capability and classification accuracy.

DJSCC set-up. We convert each trained model into a DJSCC system by inserting a communication pipeline after the third block. Then, we fine-tune the complete DJSCC

[4]The code, used to run all the experiments, can be found in the GitHub repository.

model exploring two distinct training scenarios. In the *robust* training approach, for each training sample, we randomly sample an SNR value from a uniform distribution $\mathcal{U}[-20, 20]$ dB and apply the corresponding channel noise to the transmitted features. This approach acts as a regularization technique, ensuring that each sample is processed multiple times with different SNR values during training, making the model robust against channel perturbations. In contrast, in the *noiseless* scenario, we do not apply any noise during training. However, we always apply noise during testing to evaluate model performance under variable channel conditions, regardless of the training approach used. For a comprehensive evaluation, we test each DJSCC model with five different feature compression factors r: 0.005, 0.1, 0.15, 0.25, and 0.5. These varying compression levels allow us to analyze the trade-off between transmission efficiency and performance.

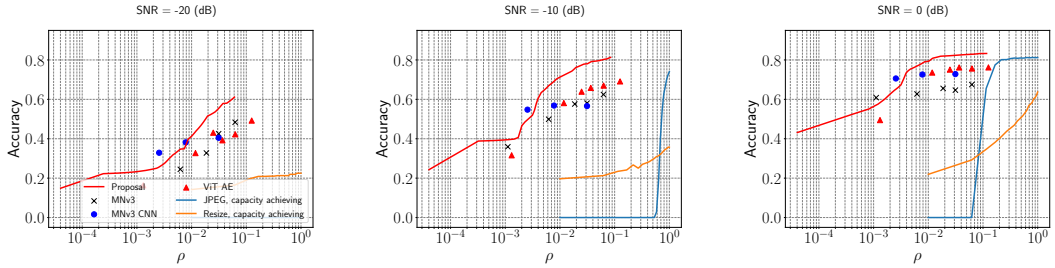
Baselines. We compare our proposed approach against two categories of baselines: neural network-based methods and digital capacity-achieving communication schemes. For our neural network comparisons, we implement three baseline models. Two are based on MobileNetV3 [261]: *MNv3*, a standard MobileNetV3 where flattened features are converted to complex symbols by the DJSCC pipeline; and *MNv3 CNN*, a variant using a convolutional DJSCC approach. For ViT, we implement *ViT AE*, a static DJSCC ViT. For all such baselines, the model is described as in Eq. (3.25), and we use the same compression factors and splitting point ($s = 3$) as described in the previous section.

Additionally, we consider two digital capacity-achieving communication baselines that compress and transmit the original image $\mathbf{x} \in \mathbb{R}^p$ directly to the edge server, where classification is performed using a pretrained ViT model. According to Shannon's capacity formula, transmitting q symbols per image over a channel with a given SNR allows for a maximum of $b_{\max} = q \log_2(1 + \text{SNR})$ bits to be reliably transmitted. If the image can be compressed to $b_{\max}$ bits or fewer, we assume perfect reception at the receiver and pass the compressed image to the ViT for classification. Otherwise, the image is considered misclassified, reflecting the system's inability to transmit it within the capacity constraint. The first baseline, *Resize*, compresses the original $H \cdot W \cdot 3$ image into an $L \cdot L \cdot 3$ image, where each pixel channel is encoded using 8 bits. The total bit cost is therefore $24L^2$. We then choose the largest integer $L \geq 1$ such that $24L^2 \leq b_{\max}$. If such L cannot be found, the image is marked as misclassified. The second baseline is *JPEG*. In this case, we select the highest JPEG quality factor that results in a compressed image size (in bits) less than or equal to $b_{\max}$. If no such quality factor can be found, the image is marked as misclassified.

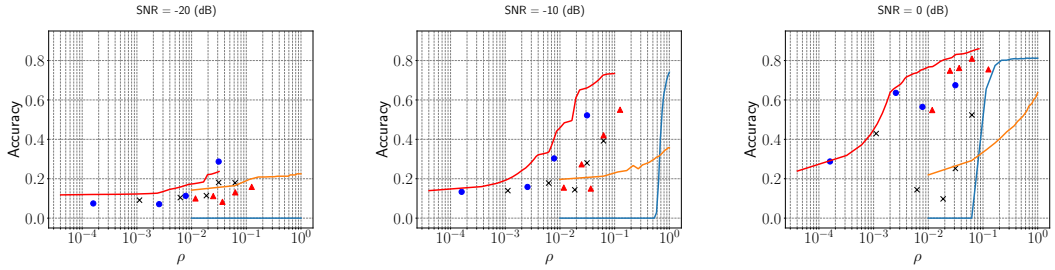
FLOPs gain of Adaptive Semantic Token Selection

In this section we evaluate our claims about the impact of discarding tokens. Figure 3.29 shows how accuracy and FLOPs gain changes when varying the budget parameter α; all results are calculated using the pre-trained models later used for the DJSCC experiments, in which the splitting point is $s = 3$.

We see that by increasing the regularization effect, the gain of FLOPs increases, to the detriment of the accuracy, as expected; the opposite is true for lower values of the same regularization weight.



(a) Accuracy curves when using robust DJSCC models.

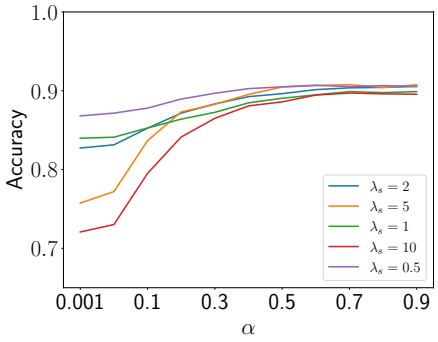


(b) Accuracy curves when using noiseless DJSCC models.

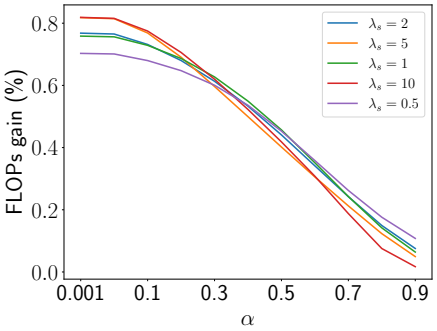
Figure 3.28. Classification accuracy versus compression ratio ρ at different (low) SNR levels, comparing the proposed method with the baselines under two training settings: (a) robust training with noisy channels; and (b) training with ideal (noiseless) channels. The legend, shown in the upper-left plot, is shared across all subfigures.

DJSCC performance with respect to noise and compression

In this paragraph, we illustrate the performance of the proposed adaptive DJSCC model in terms of compression efficiency, considering both low and high SNR regimes, and compare it against the baseline methods. To this end, in Figure 3.27 and Figure 3.28, we present the classification accuracy as a function of the compression ratio ρ, comparing the proposed method with the baselines under high and low SNR conditions, respectively. Each figure considers two training scenarios: (a) training is performed over noisy channels with the corresponding SNR, and (b) training is done assuming ideal (noiseless) channels. In both cases, testing is conducted using the SNR specified in the respective subfigure. In these figures, the results for digital capacity-achieving communication schemes are shown as continuous lines, while the neural network baselines are depicted as discrete points, since each compression ratio requires training a separate DJSCC model, resulting in different ρ values. Our proposed method, on the other hand, enables a finer and more flexible selection of the final compression ratio ρ, as a single model is trained to support multiple configurations by varying the input budget. This adaptability bridges the gap between the discrete compression ratios achievable by individual autoencoders. As a result, our method is represented by a single continuous curve, where only the best-performing configurations for each ρ value, in terms of classification accuracy, are shown. As we can notice from Figure 3.27 and Figure 3.28, our method consistently outperforms all baselines, both digital and neural, achieving up to one or two orders of magnitude improvement in compression ratio compared to digital communication schemes for a given classification accuracy. As expected, the results obtained with robust training outperform those from training under ideal noiseless conditions, particularly in



(a) How the final accuracy changes when varying α.



(b) How the FLOPs gain, in percentage over the standard model, changes when varying α.

Figure 3.29. The images show how the accuracy and the FLOPs gain, in terms of percentage respect of not discarding tokens, achieved by our proposal. The results are shown for various configuration of the training loss by changing its weighting parameters. In all experiments we fixed λ_r to 1.

low and challenging SNR regimes. Conversely, at high SNR levels, the performance of both training strategies tends to yield similar results. Actually, from (Figure 3.27), we can notice how the models trained under ideal noiseless conditions slightly outperform their robustly trained counterparts. This is likely due to the SNR augmentation range used during training (set to $[-20, 20]$ dB), which may lead to performance flattening at the high end of the SNR spectrum. The adaptability of our proposal, combined with superior performance across different SNR regimes and budget constraints, demonstrates the practical advantages of our approach in real-world communication scenarios.

Dynamic Optimization in Time-varying Channel Conditions

In this section, we analyze the results obtained using the optimization approach proposed in Section 3.3.4 for the dynamic adaptation of the selection and compression parameters $\alpha(t)$ and $r(t)$. We evaluated the proposed method across multiple scenarios. Specifically, for each threshold $\Gamma_{\text{th}} \in \{0.005, 0.0025, 0.001, 0.02, 0.05, 0.01, 0.1\}$, we search for the optimal combination of the control parameters V and μ, with search spaces defined as $V \in \{1, 10, 100, 1000, 10^4\}$ and $\mu \in \{1, 10, 100\}$. To simulate different communication scenarios, we considered several SNR (dB) distributions: $\mathcal{N}(10, 2.5)$, $\mathcal{N}(20, 2.5)$, and $\mathcal{N}(0, 2.5)$. In addition, we included two fixed SNR values: 10 dB and 20 dB. For scenarios involving statistical SNR distributions, we performed five independent optimization runs and report the average results. As baselines, we compare our proposed method against the previously introduced ViT AE and MobileNetV3 CNN models. Both baselines are trained with multiple compression factors r. For such models, the compression ratio $r(t)$ is the only tunable parameter.

To assess the performance of our approach, Figure 3.30 presents representative optimization results under both robust and noiseless training regimes. On average, our method consistently outperforms the baselines across all scenarios. While ViT-AE achieves competitive performance in certain cases, our model maintains a clear advantage, particularly when robustness to channel noise is considered (Figure 3.30, top

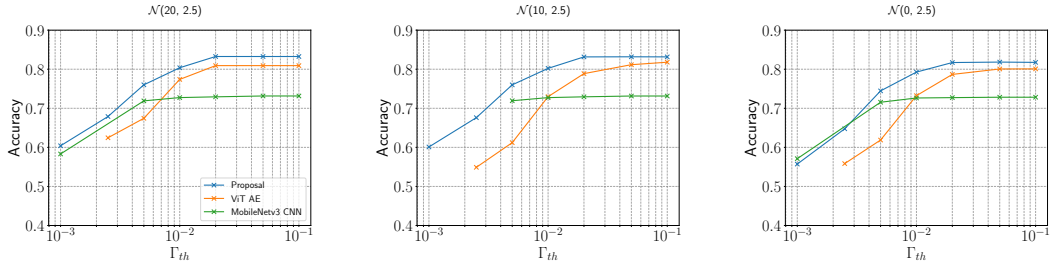
SNR (dB)	Method	Γ_{th}					
		0.0025	0.005	0.01	0.02	0.05	0.1
20	ViT AE	**62.96** - 65.68	68.05 - 70.11	**77.48** - 79.64	**80.92** - 81.49	**80.79** - **83.40**	80.97 - **84.09**
	MNv3 CNN	-	72.38 - 65.38	72.71 - 65.91	72.33 - 66.65	73.20 - 67.36	73.10 - 67.46
	Proposal	**63.41** - **72.20**	**71.14** - **76.61**	**78.93** - **81.72**	**81.15** - **83.73**	**81.07** - **84.23**	**85.15** - **85.55**
$\mathcal{N}(20, 2.5)$	ViT AE	62.34 - 65.84	67.26 - 70.67	77.06 - **79.34**	**80.92** - **81.83**	**80.54** - 82.55	80.85 - 83.25
	MNv3 CNN	-	72.10 - 65.47	72.75 - 65.74	72.65 - 66.45	73.15 - 67.31	73.16 - 67.30
	Proposal	**64.14** - **71.90**	**74.73** - **76.66**	**79.11** - **79.97**	**81.21** - **81.48**	**81.16** - **84.39**	**81.11** - **85.52**
10	ViT AE	**61.01** - 57.35	**66.90** - 63.06	77.20 - 76.41	**81.04** - **83.41**	**80.43** - **83.63**	**80.36** - 83.45
	MNv3 CNN	-	71.97 - 65.35	72.92 - 65.45	72.54 - 66.14	73.12 - 67.41	73.10 - 67.31
	Proposal	**62.88** - **70.88**	**66.23** - **72.99**	**79.31** - **79.69**	**81.10** - **83.48**	**81.07** - **84.46**	**80.82** - **85.50**
$\mathcal{N}(10, 2.5)$	ViT AE	60.65 - 57.49	66.72 - 64.53	76.82 - 77.19	**80.22** - 81.02	**80.69** - **83.90**	**80.34** - 83.26
	MNv3 CNN	-	71.83 - 65.55	72.70 - 65.52	72.97 - 66.20	73.09 - 67.24	73.00 - 67.29
	Proposal	**63.27** - **69.90**	**73.31** - **74.51**	**78.92** - **79.16**	**80.98** - **83.04**	**81.04** - **84.36**	**81.04** - **85.03**
$\mathcal{N}(0, 2.5)$	ViT AE	**55.80** - **63.80**	61.07 - **72.79**	**73.35** - **79.04**	**79.12** - **80.72**	80.02 - **81.45**	**80.22** - 82.13
	MNv3 CNN	-	71.41 - 62.85	72.63 - 64.32	72.79 - 65.58	72.95 - 67.30	72.93 - 66.95
	Proposal	**55.13** - **64.80**	**65.27** - **73.00**	**72.70** - **78.73**	**78.37** - **79.88**	**79.90** - **82.04**	**80.34** - **83.81**

Table 3.10. For each combination of SNR (in dB, drawn from a distribution or a fixed point) and Γ_{th}, the accuracy calculated on the test is shown as P - R, where P is the result of the minimization problem for noiseless communication pipelines, and R for the robust ones. Missing values correspond to failed minimization attempts (the ρ value is higher than the threshold). The best values, and others with no more than 1% accuracy difference from the best one, are highlighted in bold.

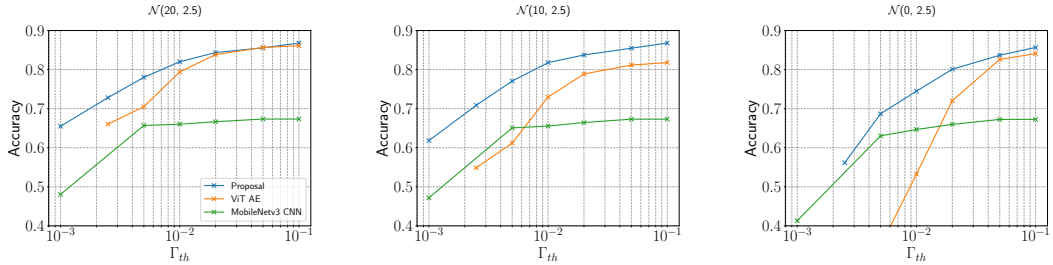
row). MobileNet, although it benefits from robust training, consistently underperforms compared to ViT-based approaches. Finally, Table 3.10 reports the test accuracy across various SNR (dB) distributions and constraint values Γ_{th}. Both robust and noiseless variants of the DJSCC models are evaluated. Overall, ViT-based models consistently achieve higher accuracy across all conditions. Notably, under stricter constraints (i.e., lower Γ_{th}), our proposed method outperforms all baselines, demonstrating its effectiveness in bandwidth-limited scenarios. As the constraint is relaxed, the performance gap between our model and ViT-AE narrows, reflecting the increased robustness to channel noise when more redundant features can be retained with lighter compression. In contrast, MobileNet models consistently underperform and often exhibit counterintuitive behavior, with accuracy degrading under robust training. This highlights their limited suitability for semantic transmission in noisy environments.

Semantic Interpretability of Token Selection

Our approach enables detailed visualization of token selection patterns across in Figure 3.31. These visualizations reveal a strong correlation between the retained tokens and semantically meaningful regions of the input, particularly under constrained computational budgets. In such settings, the model learns to prioritize tokens that convey task-relevant information, optimizing performance within limited resources. The token selection patterns highlight two notable phenomena. First, the model consistently favors tokens associated with semantically significant regions, demonstrating its ability to identify and preserve task-critical content. Second—and perhaps more intriguingly—certain tokens that may appear unimportant to human observers are consistently retained. This seemingly counterintuitive behavior can be attributed to the model's capacity to compress and redistribute information from discarded tokens into the retained ones, as



(a) Accuracy curves when using robust JSCC models.



(b) Accuracy curves when using noiseless DJSCC models.

Figure 3.30. The best results obtained for each minimization process are shown for different values of SNR (dB) distributions. On the top row, the results are obtained using robust JSCC models, while in the bottom one, the results are for noiseless ones. The legend is shared across all images.

similarly observed in [132]. This behavior indicates that the model develops sophisticated information preservation strategies that go beyond naive semantic filtering. These visualization capabilities not only offer insights into the model's internal decision-making process but also lay the groundwork for explainable-by-design AI-based communication systems.

3.3.6 Conclusion and Future Work

In this work, we introduced a novel transformer-based design for semantic edge inference, leveraging semantic token communication as a core mechanism.

We propose a transformer-based semantic token selection mechanism that dynamically identifies and transmits task-relevant input components, such as image patches. The architecture compresses selected tokens into complex-valued representations with minimal reconstruction loss, maintains robustness across diverse channel conditions using a single model, and remains compatible with standard real-valued neural networks for seamless integration with pretrained models. By reducing the number of tokens processed and transmitted, it lowers computational and communication overhead, making it suitable for resource constrained edge environments. Additionally, a stochastic optimization-based resource allocation algorithm adapts system parameters in real time to balance inference accuracy and communication cost under varying conditions, enabling efficient and adaptive semantic transmission. Supported by a robust and flexible DJSCC pipeline, our approach enables adaptive handling of communication constraints while preserving high accuracy. Our experimental results demonstrate that the proposed model consistently outperforms existing baselines, delivering superior

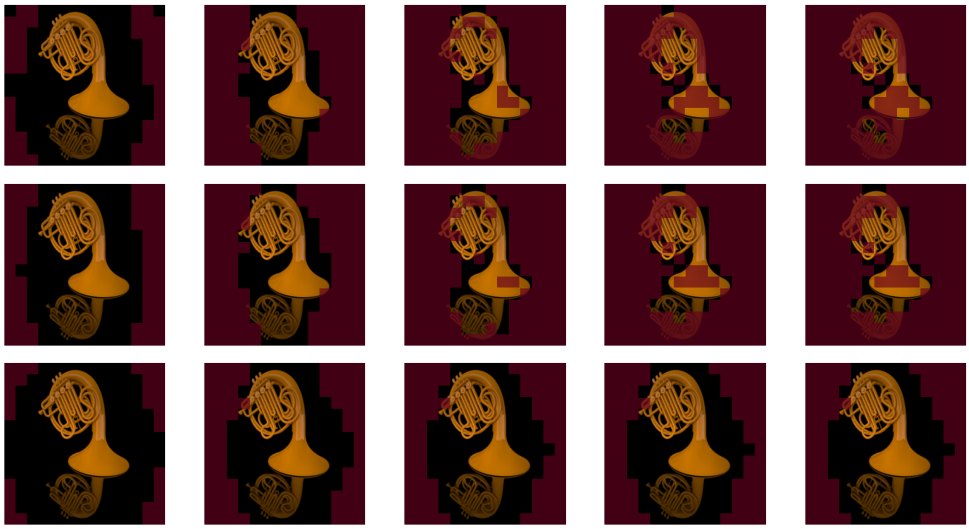


Figure 3.31. By decreasing the budget (from top to bottom), we increase the amount of discarded tokens (in red) at each block. The selected blocks, one per column, are 1, 2, 3, 5, and 12, while the selected budgets, one per row, are 0.1, 0.2, and 0.6.

accuracy across diverse operational constraints while maintaining robustness to channel impairments. A significant feature of our approach is its contribution to interpretable AI-native communication systems through the inherent transparency of the semantic token selection process. The ability to visualize and understand token selection patterns provides valuable insights into the system's decision-making process.

As future developments, we aim to extend this work across multiple data modalities, as we hypothesize that certain tasks may benefit from even more compact representations using fewer tokens. Additionally, we plan to explore the integration of trainable empty tokens, which could serve as a form of memory within the communication process—facilitating more sophisticated information storage and retrieval post-transmission. We also intend to investigate multi-user communication scenarios, where efficient resource allocation—such as bandwidth, power, and token budgets—becomes critical. In such settings, the semantic communication framework must balance the needs and priorities of multiple users while preserving overall system performance. These enhancements have the potential to further improve the flexibility, scalability, and expressiveness of semantic communication systems.

Chapter 4

Efficient Inference for Large Language Models

4.1 A simple and Effective method for KV Cache compression

Handling long contexts is desirable for large language models (LLMs), as it allows them to perform tasks that require understanding long-term dependencies [365, 189, 106, 545, 680, 582]. A key component for modelling long context is the KV Cache, which stores the keys and values of past tokens in memory to avoid recomputing them during generation. However, processing long-context inputs often results in a high decoding latency since it requires repeatedly reading a potentially large KV Cache from high-bandwidth memory (HBM) to the streaming multiprocessor (SM) during decoding [188]. Consequently, the practical deployment of LLMs is frequently hindered by hardware limitations. To address the issue of KV Cache growth, various KV Cache compression methods have been proposed. These methods can be broadly categorised into trainable approaches, which involve modifications to the model architecture [8], or fine-tuning regime to inherently manage KV Cache size [421], and non-trainable approaches, which apply post-hoc compression techniques to reduce the cache footprint without altering the underlying model [348, 676, 196]. While these methods have shown promise, they often involve complex algorithms or significant computational overhead, limiting their practicality; for example, post-hoc compression algorithms usually evict KV pairs based on attention scores, which is not compatible with FlashAttention [130] and thus prevents their applications in modern LLMs inference systems.

We show that, surprisingly, the L_2 norm of cached keys has a high correlation with attention scores. More specifically, we observe that a low L_2 norm of a key embedding usually leads to a high attention score during decoding. Based on this observation, we propose a simple and highly effective strategy for KV Cache compression: *keeping in memory only the keys with lowest L_2 norm, and the corresponding values*. Unlike many existing methods, our heuristic can be applied off-the-shelf to any transformer-based decoder-only LLM without the need for additional training or significant modifications. More importantly, our method estimates the influence of cached key-value pairs without the need to compute the attention scores. Therefore, unlike other compression methods [254, 348], it can be easily integrated with the popular FlashAttention [130].

Our experimental results demonstrate that this heuristic allows maintaining model

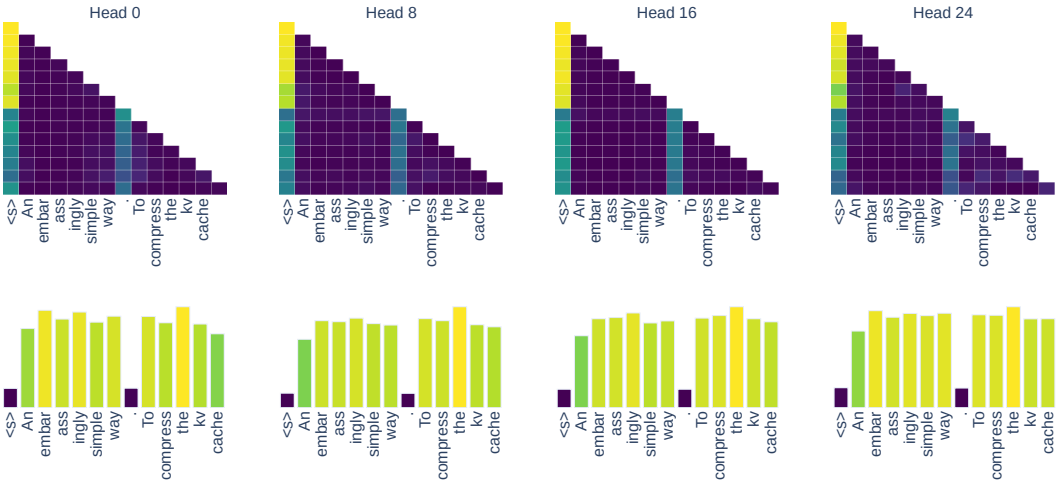


Figure 4.1. Five heads at layer 9 of Llama2-7b. Attention score (top) and L_2 norm (bottom) are highly correlated. We observe similar patterns across most layers and for a wide range of inputs. More examples provided in Section 4.1.10

performance in language modelling tasks and in tasks that require the model to store and retrieve the most critical information, such as passkey retrieval [408] and needle-in-a-haystack tasks [290].

4.1.1 Background on LLM Inference

In transformer-based LLMs, the input sequence is represented as a tensor $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the sequence length and d is the token embedding dimension. Each x_i corresponds to an embedding of a token in the sequence. The tensor $\mathbf{X}$ is processed by a series of transformer blocks, each composed of a multi-head self-attention and a feed-forward layer.

Given an input $\mathbf{X} \in \mathbb{R}^{n \times d}$, the multi-head attention mechanism performs multiple attention operations in parallel, allowing the model to attend to information from different representation subspaces. It does so by first computing three projections: the query, key, and value matrices, denoted as $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, respectively. These are obtained by linear transformations of the input $\mathbf{X}$:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V, \tag{4.1}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_k}$ are learned projection matrices, and d_k is the dimensionality of the queries and keys. Next, the output is computed using the scaled dot-product attention. The attention output is calculated as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{4.2}$$

In the multi-head attention mechanism, this process is repeated h times, each with different learned projections $\mathbf{W}_Q^{(i)}, \mathbf{W}_K^{(i)}, \mathbf{W}_V^{(i)}$ for each head h, resulting in H separate

attention outputs. These outputs are concatenated and projected back to the original dimension d using a final learned matrix $\mathbf{W}_O \in \mathbb{R}^{hd_k \times d}$:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)\mathbf{W}_O \tag{4.3}$$

where each attention head head_h is defined as $\text{head}_h = \text{Attention}(\mathbf{Q}^{(h)}, \mathbf{K}^{(h)}, \mathbf{V}^{(h)})$.

KV Cache During autoregressive inference, where tokens are generated sequentially, the model has to compute the attention distributions over all previously generated tokens at each step. Without optimisations, this would involve recalculating the key ($\mathbf{K}$) and value ($\mathbf{V}$) projections for every past token at each new step. The KV Cache addresses this inefficiency by storing the key and value projections for each token after they are first computed. Instead of recalculating these projections for past tokens, the model retrieves the cached $\mathbf{K}$ and $\mathbf{V}$ values during subsequent inference steps.

When generating a new token at time step t, the attention computation is performed as:

$$\text{Attention}(\mathbf{Q}_t, [\mathbf{K}_{1:t-1}; \mathbf{K}_t], [\mathbf{V}_{1:t-1}; \mathbf{V}_t]) \tag{4.4}$$

where $[;]$ denotes concatenation along the sequence dimension, and $\mathbf{K}_{1:t-1}$ and $\mathbf{V}_{1:t-1}$ are retrieved from memory. The key $\mathbf{K}_t$ and value $\mathbf{V}_t$ for the current token are computed normally.

The KV Cache can significantly reduce computational costs by avoiding redundant calculations. However, storing the cached key and value matrices for every token in the sequence incurs substantial memory usage, which grows linearly with the sequence length. For a model with L layers, H attention heads, and a sequence length of n, the total memory required is $L \times H \times n \times d_k \times 2\times$, where the factor of 2 accounts for both the key and value matrices and *precision* represents the number of bytes used to store each value in the memory, typically corresponding to the bit-width of the data type (e.g., 16 bits for half-precision or 32 bits for single-precision floating point).

Though the KV Cache improves the computational efficiency, it requires repeatedly reading potentially large KV Cache from high-bandwidth memory to the streaming multiprocessor during decoding. To address this, recent works [676, 254, 348, 375] have proposed compressing the KV Cache to reduce memory usage.

4.1.2 Analysis of the Attention Distributions

We first examine the attention scores on the language modelling task for a range of popular LLMs. By analysing the key embeddings and the attention distribution, we observe that key embeddings with low L_2 norm are often associated with higher attention scores. In Figure 4.1, we provide an example using Llama-2-7b [574], where the columns represent different heads, the first row presents the attention distribution over the KV pairs, and the second row presents the L_2 norm of each key embedding. We observe that the tokens with high attention scores, such as "<s>" and ".", have significantly lower L_2 norm values than others. While [623] already observed peaked attention distributions for specific tokens, and [133] pointed out the influence of high L_2 norm hidden states on attention maps, we are the first, to the best of our knowledge, to point out the correlation between the L_2 norm of the *key embeddings* and attention score. Based on

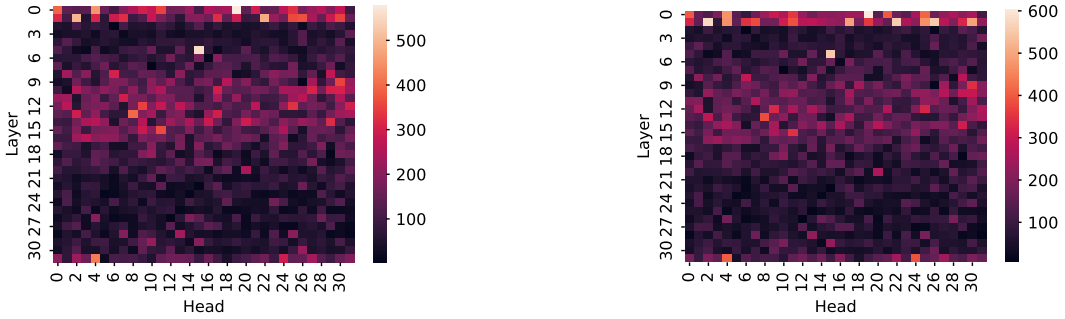


Figure 4.2. ALR, as defined in Equation (4.7), for each head and layer in Llama2-7b (left) and Llama2-7b-32k long context model (right). A lower value means a higher correlation between L_2 norm and attention score.

our observation, we consider the following research question: can we compress the KV Cache based on the L_2 norm of the key embeddings?

An intuitive way to estimate the influence of compressing the KV Cache is by examining the attention scores that are dropped due to the compression. In the following, we formally define this influence.

Given a prompt consisting of n tokens $(x_1, x_2, ..., x_n)$, the LLM first encodes them into a KV Cache— this step is referred to as the *pre-filling phase*. Then, the model autoregressively generates the next token x_{n+1}. When performing KV Cache compression, some key-value pairs may be dropped and thus cannot be attended to. We define the attention loss caused by the compression as the sum of the attention scores associated with the dropped KV pairs:

$$\mathcal{L}_{l,h}^{m} = \sum_{p \in D_{l,h}} a_{l,h,p}, \tag{4.5}$$

where $a_{l,h,p}$ is the attention score of the p-th token in the layer l, head h. In Equation (4.5), $D_{l,h}$ denotes the positions of m pairs of dropped KV, $|D_{l,h}| = m$, which depends on the compression method. An ideal compression algorithm aims to drop the KV pairs with the lowest attention scores, which will have less impact on the output. However, such attention scores are unavailable for a compression algorithm since it needs x_{n+1} to query the full KV Cache in advance. Instead, we drop KV pairs with the highest L_2 norm in key embeddings and use attention loss caused by ideal compression as the reference:

$$\mathcal{Y}_{l,h}^{m} = \mathcal{L}_{l,h}^{m} - \mathcal{L}_{l,h}^{m,ref}, \tag{4.6}$$

where $\mathcal{L}_{l,h}^{m,ref}$ is the reference attention loss, and $\mathcal{Y}_{l,h}^{m}$ is a non-negative value. A lower $\mathcal{Y}_{l,h}^{m}$ indicates a lower difference and thus a higher correlation between the attention score and the L_2 norm. To measure the overall difference between ideal attention score-based compression and L_2 norm-based compression, we sum up the $\mathcal{Y}_{l,h}^{m}$ over different numbers of compressed KV pairs:

$$\mathcal{Y}_{l,h} = \sum_{m=1}^{n} \mathcal{Y}_{l,h}^{m}. \tag{4.7}$$

We name the $\mathcal{Y}_{l,h}$ as ALR, which denotes the attention loss (Equation (4.5)) for a compression method using the ideal attention loss as reference. In Figure 4.2, we plot

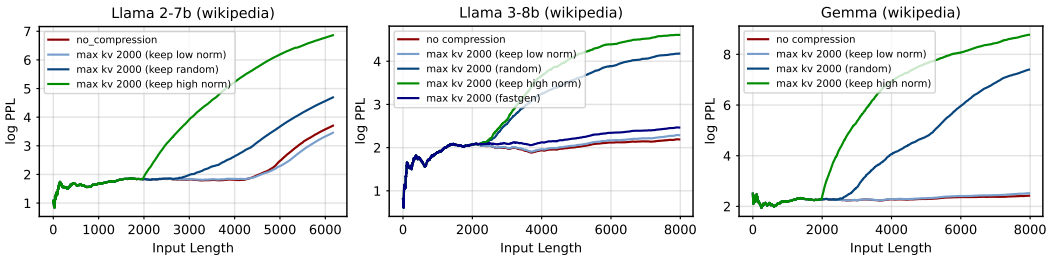


Figure 4.3. Perplexity for Llama 2-7b, Llama 3-8b and Gemma on language modelling task on wikipedia dataset.Additional results on coding dataset are available in Section 4.1.6

the γ across layers and heads. We observe that heads in the first two layers and some middle layers around the 12th layer have relatively high γ values. The heads in other layers have lower γ values, indicating a high correlation between L_2 norm and attention score.

By leveraging this correlation, we can compress the KV Cache based on the L_2 norm of key embeddings. Optionally, we can skip the compression at the layers with low correlation. We show ablation experiments skipping layers in Section 4.1.6.

4.1.3 Experiments

We evaluate our method on language modelling and two long-context modelling tasks, i.e., needle-in-a-haystack and passkey retrieval. In addition, we test on tasks from LongBench [672], specifically devised to evaluate the model's long context abilities. Based on the observation supported by Figure 4.2, the heads in the first two layers usually have a low correlation between L_2 norm and attention score, so we do not perform compression on these layers as default. We conduct experiments to investigate the impact of compression on different layers in Section 4.1.6.

Language Modelling Tasks For language modelling, we let the KV Cache grow until a specific pre-defined length and subsequently start to discard the tokens with the highest L_2 norm. We show in Figure 4.3 that evicting even up to the 50% of KV Cache does not impact perplexity. Perplexity increases, as expected, once we exceed the pre-training context length. We show more results, including next token accuracy in Section 4.1.6. To further verify that keys with low L_2 norm capture significant information, we test other eviction strategies, i.e. keeping tokens with highest L_2 norm and keeping random tokens. It is clear from Figure 4.3 that discarding tokens with low L_2 impairs performance, even more so than random discarding, thus highlighting the importance of these low L_2 norm keys.

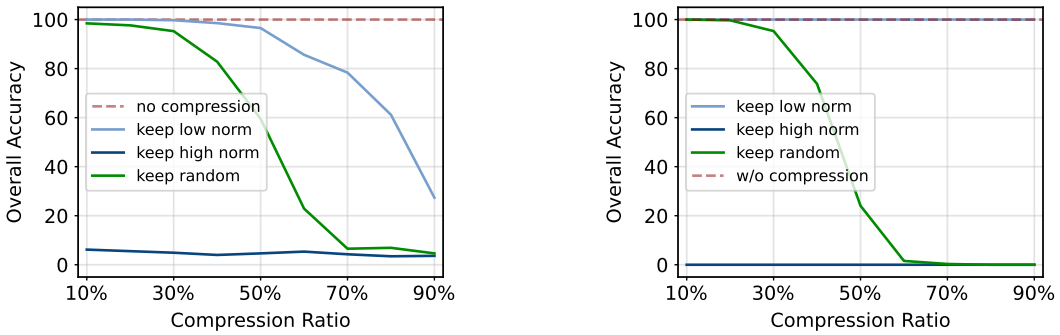
Pressure Test on Long-Context Tasks The needle-in-a-haystack task [290] and passkey retrieval task [408] are two synthetic tasks that are widely used to pressure test the long-context modelling capability of LLMs. In both tasks, the model needs to identify and retrieve the important information from a long context to generate correct answers. Thus, these tasks test the compression method's ability to keep important KV pairs and drop redundant ones.

In Figure 4.4a and Figure 4.4b, we present the experimental results of Llama-2-7b-80k [189]. We analyse additional models in Section 4.1.7. As shown in Figure 4.4a, the model can preserve its performance on the needle-in-a-haystack task while compressing 30% of the KV Cache, and maintain 99% accuracy when compressing 50% of the KV Cache. Additionally, the model can achieve 100% accuracy on the passkey retrieval task even when compressing 90% of the KV Cache, as shown in Figure 4.4b.

Moreover, we compare other eviction strategies, like keeping KV pairs with low L_2 norm, keeping KV pairs with high L_2 norm, and keeping random KV pairs. In Figure 4.4a and Figure 4.4b, we observe that the model cannot answer correctly when keeping only high L_2 norm KV pairs, obtaining near zero and zero accuracy on the needle-in-a-haystack and passkey retrieval tasks, respectively. When we randomly compress the KV Cache, the performance decreases significantly faster than keeping low L_2 norm KV pairs. The above analysis indicates that KV pairs with low L_2 norm are critical to generating the correct answer and thus contain important information.

Experiments on LongBench Additionally, we evaluate on LongBench [672]. We test on several subsets, including NarrativeQA [304], Qasper [135], HotpotQA [646], 2WikiMQA [253], and QMSum [682]. We report the results for the recently released long context Llama3.1 and Llama 2-7b 80k in Figure 4.5. In addition, we show the complete per-subset results in Section 4.1.7. The experimental results show that compressing the KV Cache with low L_2 norm only introduces a small accuracy decrease even when compressing 50% KV Cache, while compressing KV Cache with high L_2 norm results in almost zero accuracy.

Comparison with FastGen We use FastGen [254], a popular method for KV Cache compression, as a baseline for assessing the effectiveness of our method. It is important to note that, like the majority of methods in the literature, FastGen utilises attention scores, which makes it incompatible with the popular FlashAttention [130], thereby limiting its efficiency and usability. For a fair comparison, we implement FastGen without using the attention scores, i.e., we only consider local, punctuation and special tokens. We perform experiments on language modelling with the Llama3 model [399]. Our method still outperforms FastGen with up to 50% KV Cache eviction. We show the



(a) Accuracy on the needle-in-a-haystack.

(b) Accuracy on the passkey retrieval task.

Figure 4.4. Overall accuracy of llama-2-7b-80k on the needle-in-a-haystack task passkey retrieval task.

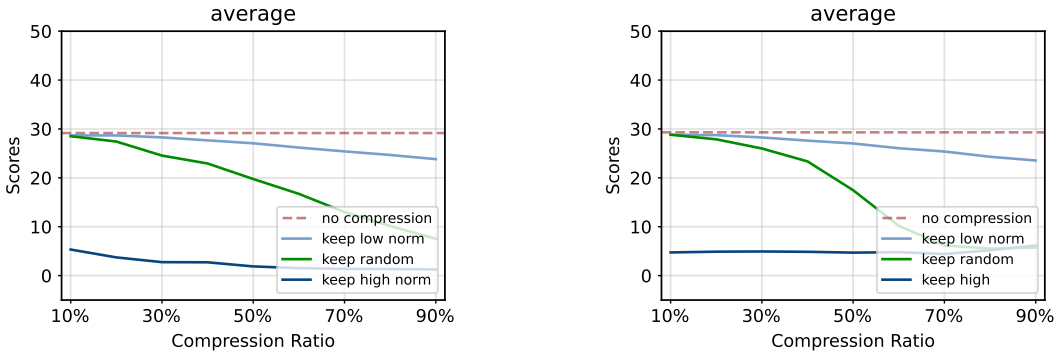


Figure 4.5. Overall scores on LongBench [672] of Llama3.1-8b (left) and llama-2-7b-80k (right) for different compression ratios ranging from 0% to 90%.

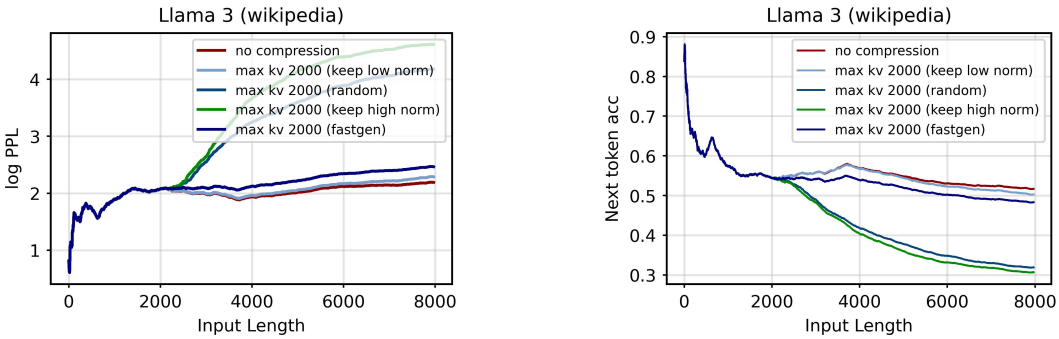


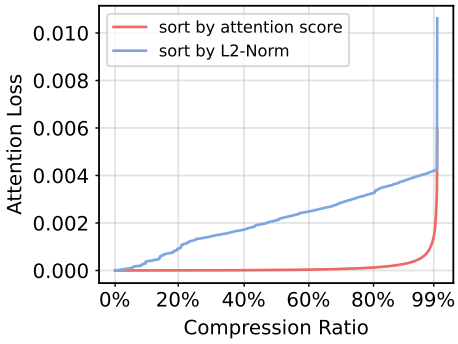
Figure 4.6. Perplexity and next token accuracy of Llama3-8b on the wikipedia dataset when compared to FastGen [254] (only local, special and punctuation tokens).

results in Figure 4.6.

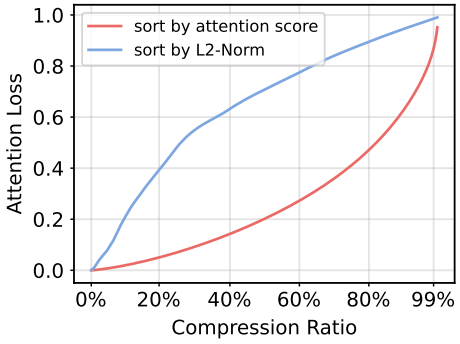
4.1.4 Analysis

Attention score loss when using L_2 norm We discuss further the correlation between L_2 norm and attention scores. We already displayed in Figure 4.2 the L_2 norm and attention correlation across heads and layers using the original Llama2-7b and the long context Llama2-7b-32k and Llama2-7b-80k. We can see that patterns are quite consistent across all the models. To better visualise how correlation varies across different heads, in Figure 4.7, we only consider two heads from layer 10 and layer 0 and show the ALR from Equation (4.5). As expected, we see that in layer 0, the difference is larger due to a lower correlation.

Relationship between embedding and L_2 norm So far, we have identified a correlation between the L_2 norm of token key embeddings and the corresponding attention scores. This observation, while primarily empirical, it offers a direction for further explorations. Our investigation into the distribution of key embeddings revealed that tokens with lower L_2 norm tend to exhibit *sparse activations* with only a few dimensions showing significantly high values, while the majority of the dimensions remain near zero. This pattern suggests that the embeddings of these tokens are not fully utilising the available vector space, focusing their activations on a narrow subset of dimensions. Figure 4.8



(a) Layer-7 Head-10, high correlation between attention score and L_2-Norm.



(b) Layer-0 Head-0, low correlation between attention score and L_2-Norm.

Figure 4.7. Attention loss of ideal compression and L_2 norm-based compression in Llama-2-7b-80k. The x-axis represents the compression ratio; the y-axis represents the attention loss (defined by Equation (4.5)) The results average over 1024 chunks on Wikipedia, with a length of 1024.

illustrates several examples of such tokens, highlighting the difference between tokens with high and low L_2 norm.

Interestingly, this sparsity aligns with the concept of "sink" tokens, as identified in previous studies [623]. These tokens capture a direction in the embedding space such that many queries align closely with it, leading to increased attention scores for these tokens. Specifically, when the key embeddings of certain tokens are dominated by a limited set of dimensions, they create a focal point, attracting a wide range of queries – regardless of their individual content – and amplifying their attention weights.

We hypothesise that the lower L_2 norm reflects a partial use of the available embedding space, leading to increased attention for these tokens. To examine this hypothesis, we zeroed out the dimensions responsible for the peaked activations in low-norm key embeddings and observed significant changes in attention maps (Figure 4.9). In contrast, altering random dimensions did not produce the same effect, highlighting the

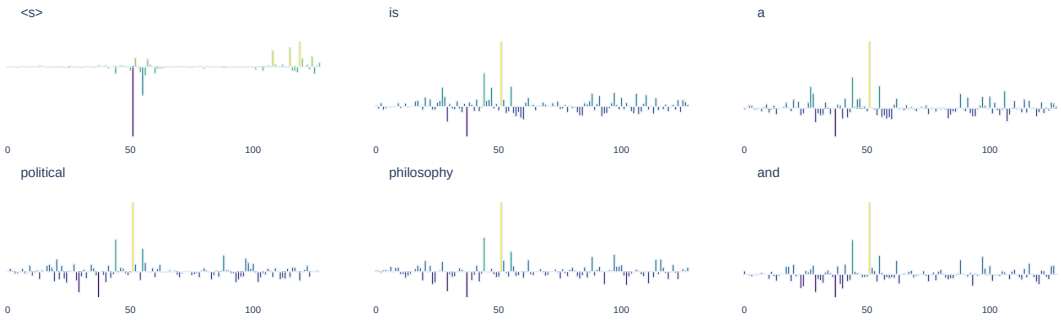


Figure 4.8. Key projections of the bos token $< s >$ vs other tokens. Each value represents the activation in a specific dimension for the embedding of the key projection. We found similar patterns across almost all heads and layers and in multiple texts. Only a few peaked activations (~ 50, ~ 56 and ~ 120) control the attention mechanism (see Figure 4.9). More plots like this in Section 4.1.11

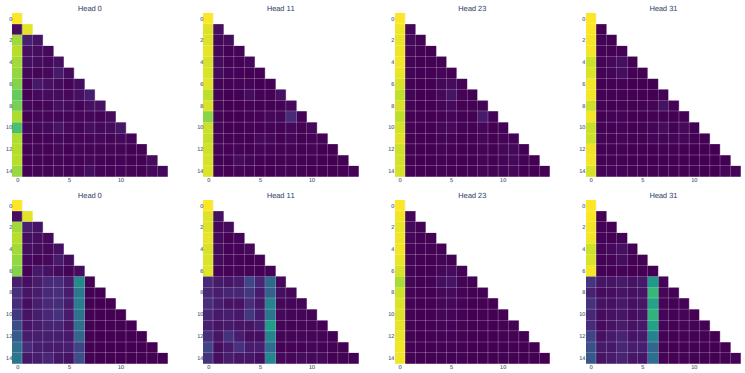


Figure 4.9. How the attention maps change if we set to zero a random activation (top) vs the specific peaked activations in the keys (bottom). We are setting the values at iteration 5.

importance of these specific dimensions. This finding suggests that the L_2 norm may serve as a proxy for the extent to which an embedding utilises the available vector space and, consequently, the degree to which it influences attention. Lower L_2 norm appears to correspond to embeddings that drive disproportionately high attention values due to their alignment with a common direction. [85] offers additional insight into this phenomenon, suggesting that attention sinks engage with other tokens through a "dark" subspace within the embedding space.

4.1.5 Related Work

Recently, various long-context LLMs, such as Gemini-Pro-1.5 [497], Claude-3 [19], and GPT4 [2], have shown the promising capability to process hundred thousands of tokens in the context. The increased number of input lengths results in a high decoding latency; thus, there has been a growing interest in speeding up the decoding with long contexts. Some works propose efficient memory management strategies to reduce the IO time overheads, e.g., PageAttention [318], Infinite-LLM [357] and vAttention [473]. Another line of research focuses on compressing the KV Cache to improve efficiency. DMC [421] compresses KV Cache by dynamically merging tokens while requiring expensive continual pre-training. For fine-tuning free compression strategy, H2O [676] identifies important KV pairs by leveraging the attention scores from all queries, FastGen [254] leverages the different attention patterns in different heads for compression, and SnapKV [348] selects KV pairs based on attention scores from user's query. Unlike these works, our method only utilises the L_2 norm of embedding for compression *without leveraging the attention information*, and to the best of our knowledge, we are the first to find that the influence of a KV pair can be determined by L_2 norm. Previous work [133] finds the hidden states with high L_2 norm usually aggregate more important and global information. On the other hand, our findings indicate that a low L_2 norm of key embedding generally results in a high attention score. Concurrently to this work, [232] uses the L_1 norm of values in the KV Cache and attention scores for compression.

4.1.6 More results on Language modelling task

In the following, we show results when performing compression only on layers that show a lower correlation between L_2 norm and attention score. We show in Fig. 4.13 that for language modelling tasks, the different layer drop has little impact on final accuracy and perplexity. The difference becomes significant only when the KV Cache is pruned to retain only one thousand pairs. All experiments are averaged over 50 chunks from English Wikipedia.

4.1.7 More Results on Long-Context Modelling Tasks

In addition to llama-2-7b-80k [189], we test the compression method using llama-2-7b-longlora-32k-ft [106] on the needle-in-a-haystack and passkey retrieval tasks. As shown in Fig. 4.15a, we can see that compressing 30% of KV Cache only results in a slight performance degradation on the needle-in-a-haystack task. We also observe that the performance even increases slightly when we compress 10% of KV Cache. In figure Fig. 4.15b, we observe that the llama-2-7b-longlora-32k-ft maintains 100% performance when compressing 80% of KV Cache and only as a slight decrease when compressing 90% of KV Cache. Furthermore, the model fails to generate correct answers if we compress KV pairs with low L_2 norm and keep high L_2 norm ones. The evaluation results of llama-2-7b-longlora-32k-ft are consistent with the llama-2-7b-80k, which further indicates the effectiveness of compressing KV Cache using L_2 norm.

4.1.8 Analysis of Skipped Layers

As shown in Fig. 4.2, we find heads in the first two layers and the middle layers have a relatively low correlation between attention scores and L_2 norm. Thus, we conduct experiments to analyse the impact of skipping layers that have a low correlation for compression. As shown in Fig. 4.16a and Fig. 4.16c, we observe that only skipping the first layer (layer-0) decreases the performance on the needle-in-a-haystack task significantly. We can see that skipping the first two layers (layer-0,1) has a similar performance compared to skipping the first three layers (layer-0,1,2). Furthermore, as shown in Fig. 4.16b and Fig. 4.16d, only skipping the first layer can result in significant performance degradation. We also find that the compression ratio is not proportional to the overall accuracy of models in the passkey retrieval task when we compress the first layer, where the accuracy shows a U-shape curve regarding the compression ratio.

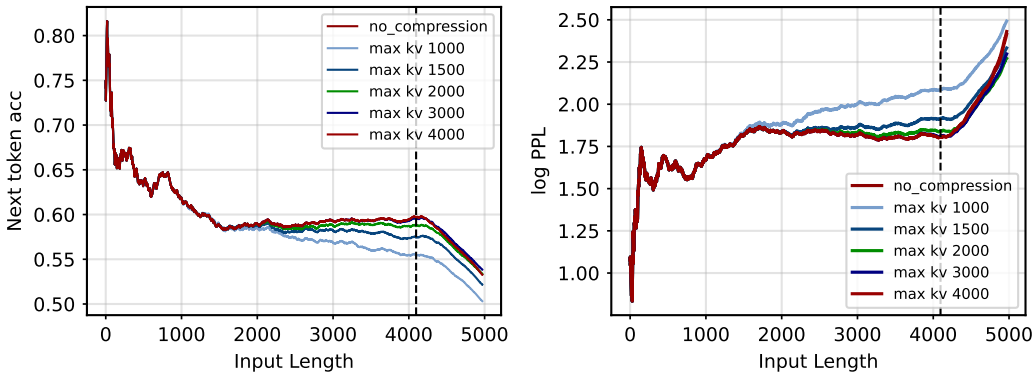


Figure 4.10. Results on language modelling task when skipping the first layer.

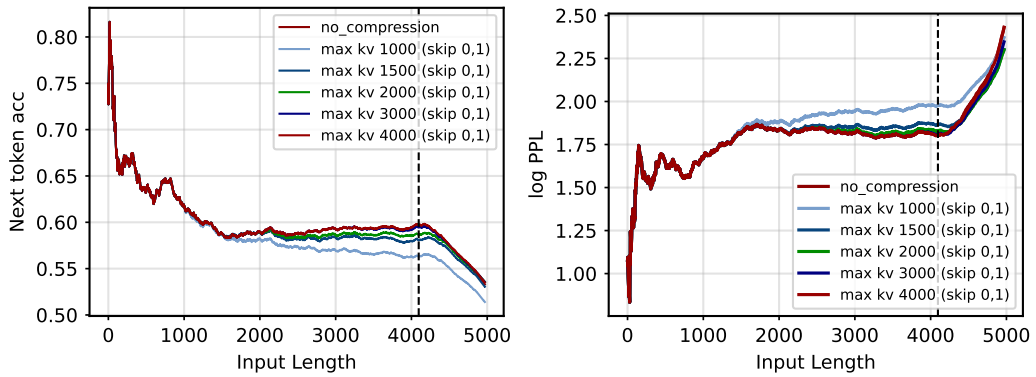


Figure 4.11. Results on language modelling task when skipping the first two layers.

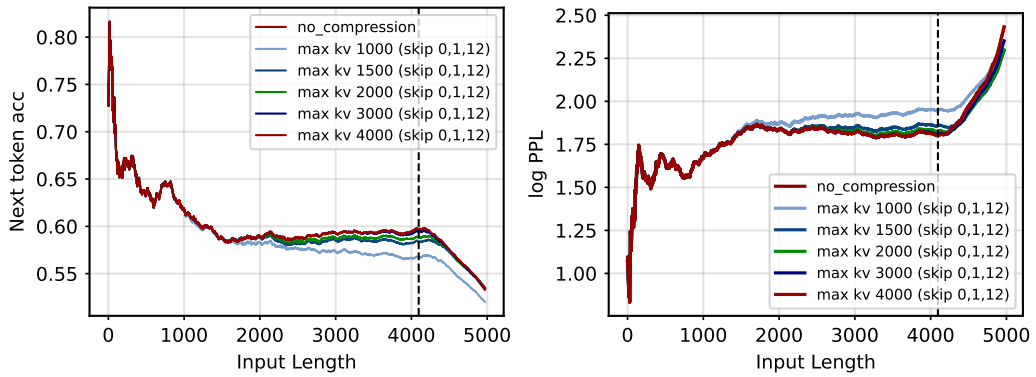
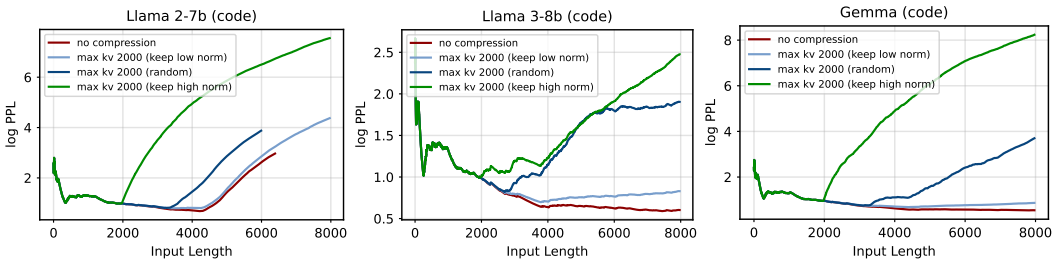
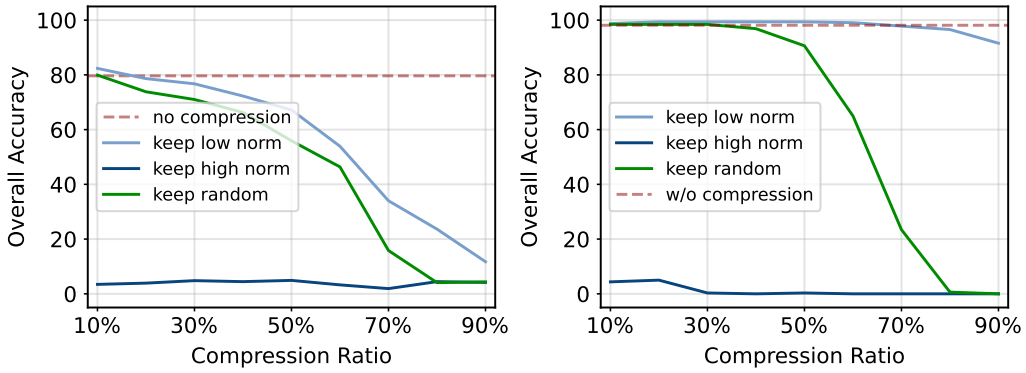


Figure 4.12. Results on language modelling task when skipping layers 0,1 and 12.

Figure 4.13. Skipping compression at different layers with Llama2-7b

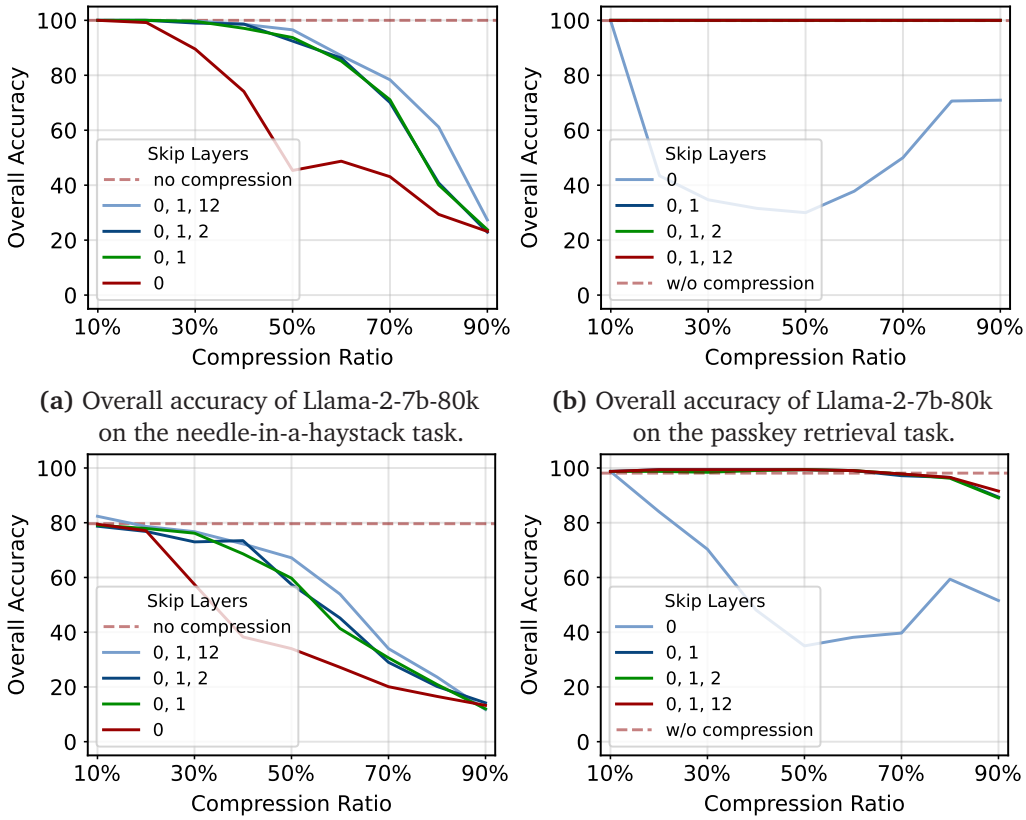




(a) Overall accuracy of Llama-2-7b-longlora-32k-ft on the needle-in-a-haystack task.

(b) Overall accuracy of Llama-2-7b-longlora-32k-ft on the passkey retrieval task.

Figure 4.15. Evaluation results of Llama-2-7b-longlora-32k-ft on the needle-in-a-haystack and passkey retrieval tasks.



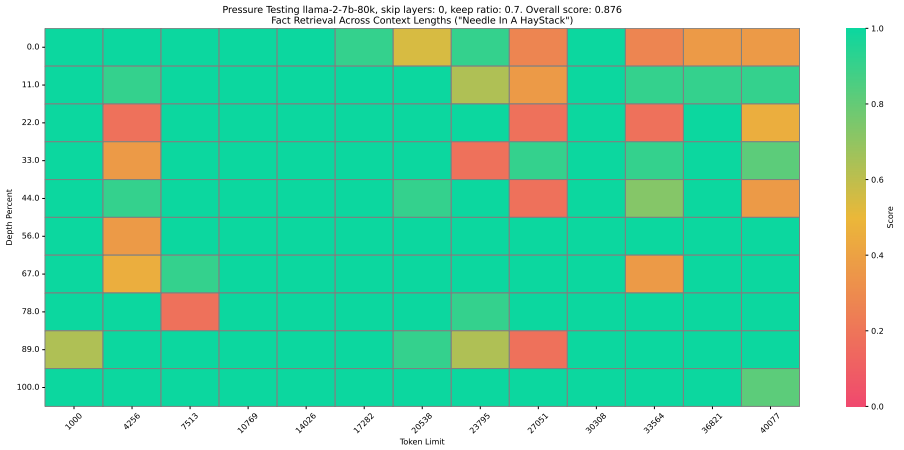
(a) Overall accuracy of Llama-2-7b-80k on the needle-in-a-haystack task.

(b) Overall accuracy of Llama-2-7b-80k on the passkey retrieval task.

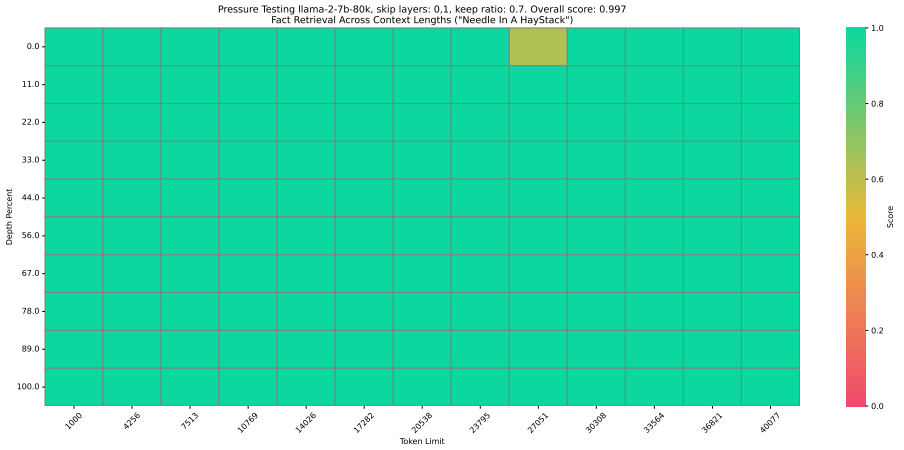
(c) Overall accuracy of Llama-2-7b-longlora-32k-ft on the needle-in-a-haystack task.

(d) Overall accuracy of Llama-2-7b-longlora-32k-ft on the passkey retrieval task.

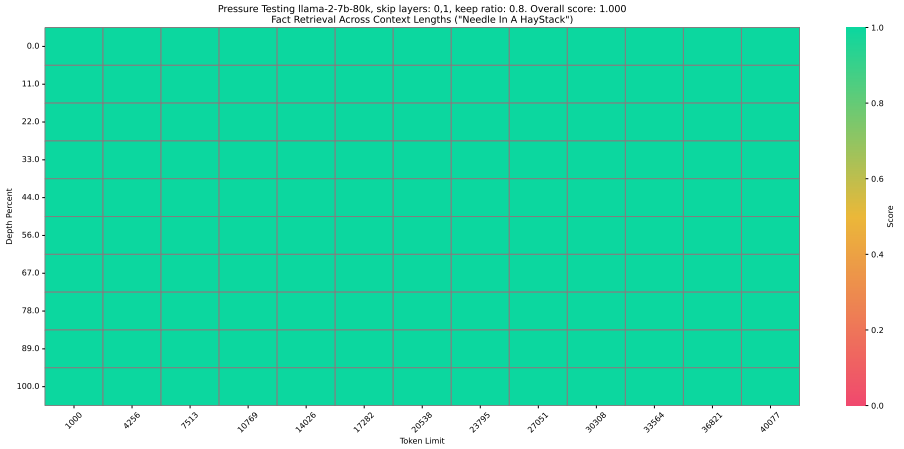
Figure 4.16. Analysing of skipping different layers for compression.



(a) Llama-2-7b-80k, skip layer-0, compression ratio 30%

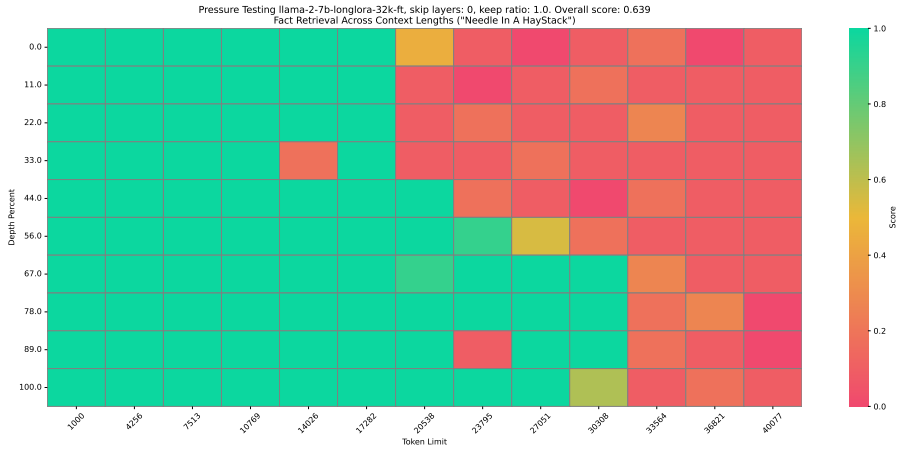


(b) Llama-2-7b-80k, skip layer-0 and layer-1, compression ratio 30%

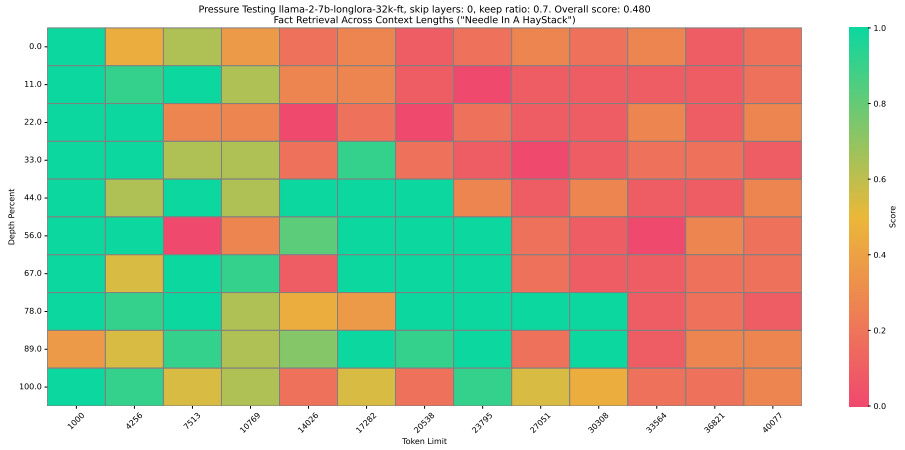


(c) Llama-2-7b-80k, skip layer-0 and layer-1, compression ratio 20%

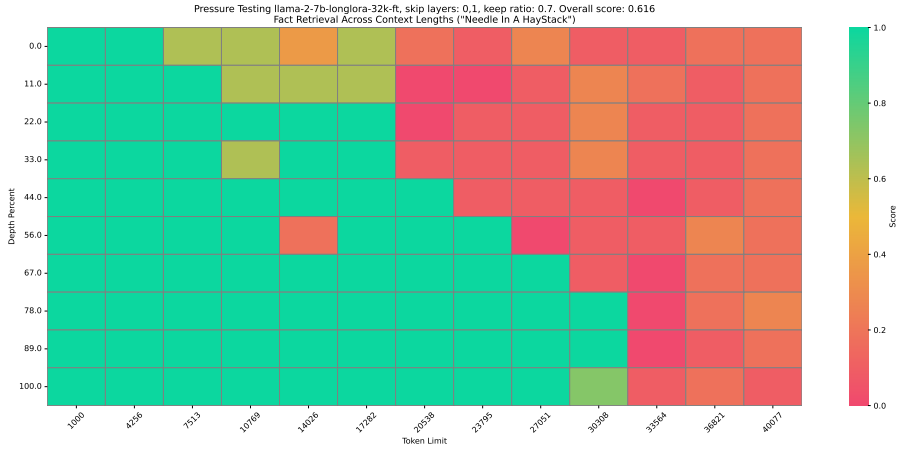
Figure 4.17. Detailed results of Llama-2-7b-80k on the needle-in-a-haystack task.



(a) Llama-2-7b-longlora-32k-ft, without compression

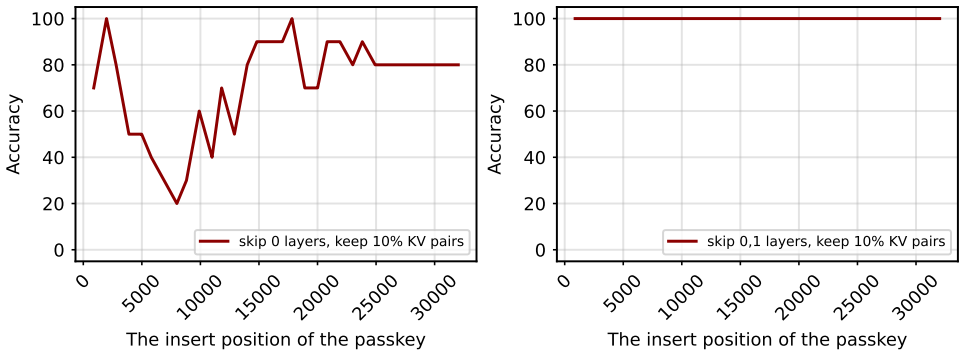


(b) Llama-2-7b-longlora-32k-ft, skip layer-0, compression ratio 30%



(c) Llama-2-7b-longlora-32k-ft, skip layer-0 and layer-1, compression ratio 30%

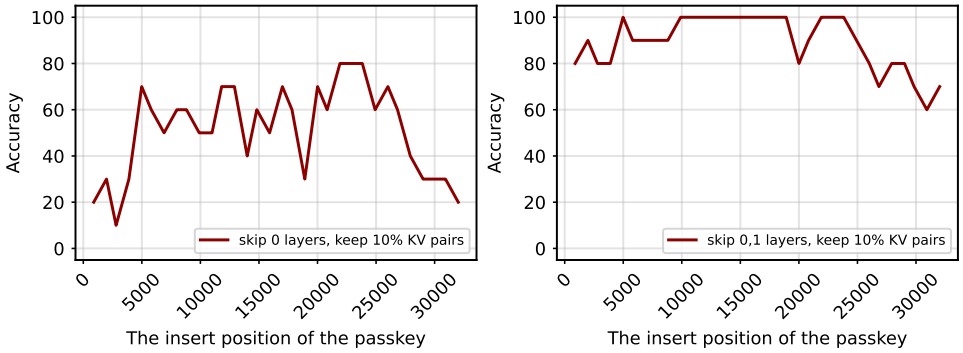
Figure 4.18. Detailed results of Llama-2-7b-longlora-32k-ft on the needle-in-a-haystack task.



(a) Llama-2-7b-80k, skip layer-0, compression ratio 90%

(b) Llama-2-7b-80k, skip layer-0 and layer-1, compression ratio 90%

Figure 4.19. Accuracy on the passkey retrieval. The x-axis presents the position of the passkey, and the y-axis presents the accuracy.



(a) Llama-2-7b-longlora-32k-ft, skip layer-0, compression ratio 90%

(b) Llama-2-7b-longlora-32k-ft, skip layer-0 and layer-1, compression ratio 90%

Figure 4.20. Accuracy on the passkey retrieval. The x-axis presents the position of the passkey, and the y-axis presents the accuracy.

4.1.9 Longbench Evaluation

In this section we show detailed results from the LongBench dataset [672]. In Figure 4.21 we show results for Llama2-80k, while in Figure 4.22 we show results for the long context model Llama3.1-8b.

4.1.10 More Visualizations

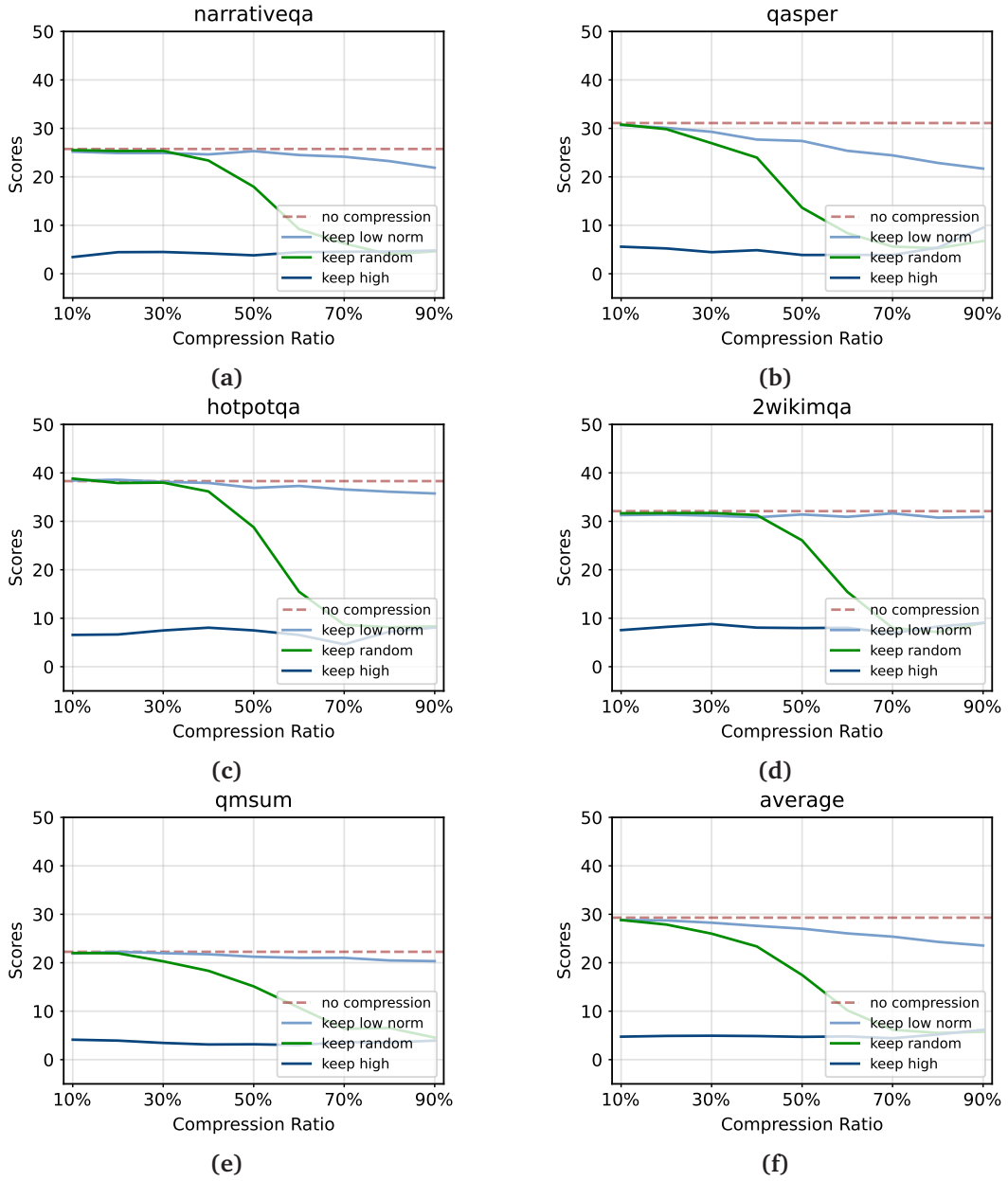


Figure 4.21. Evaluation results of Llama-2-7b-80k on long context tasks from Longbench, including narrativeqa and qasper, hotpotqa, 2wikimqa, and qmsum.

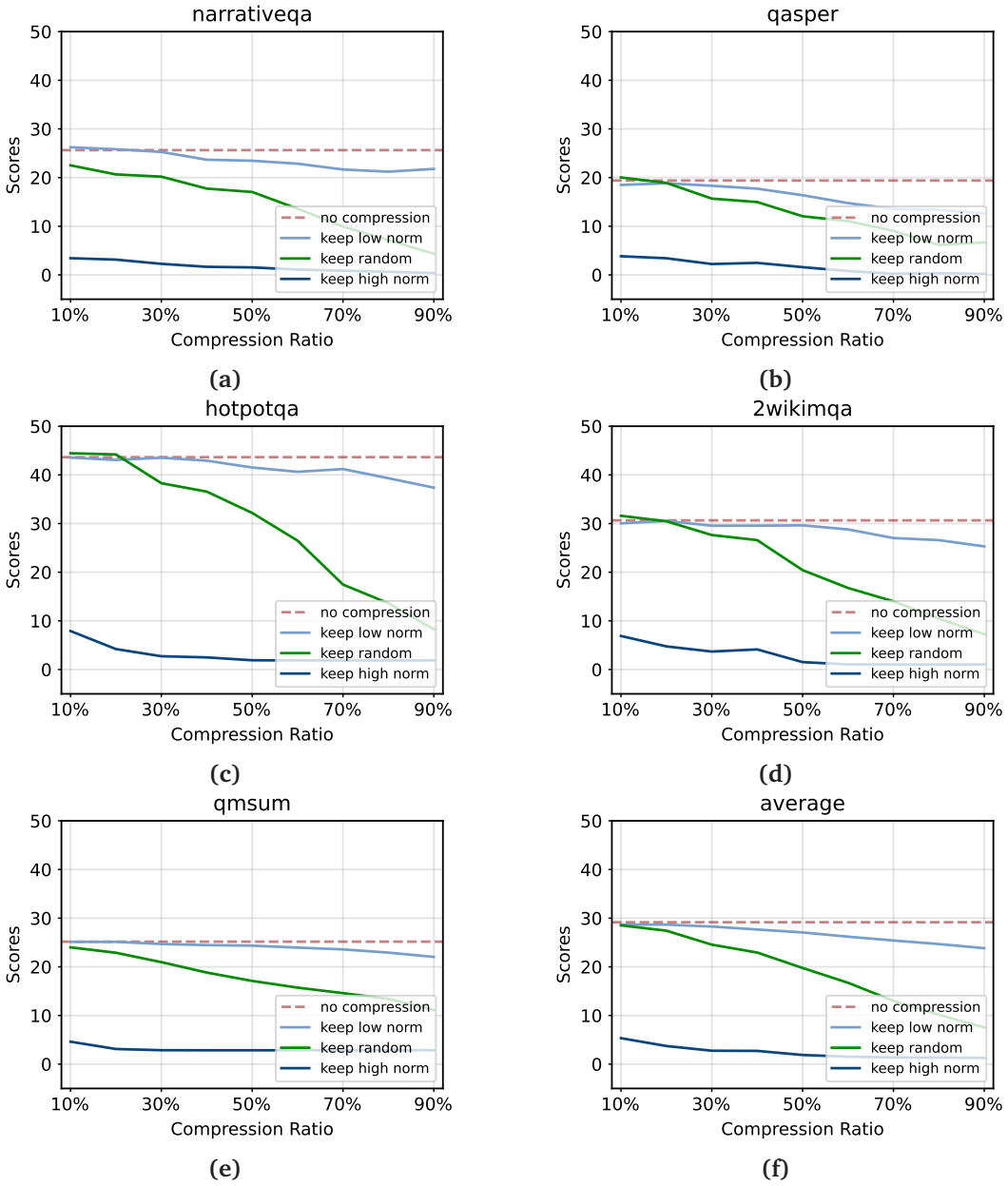


Figure 4.22. Evaluation results of Llama-3.1-8B on long context tasks from Longbench, including narrativeqa and qasper, hotpotqa, 2wikimqa, and qmsum.

Figure 4.23. Attention maps in Llama2-7B

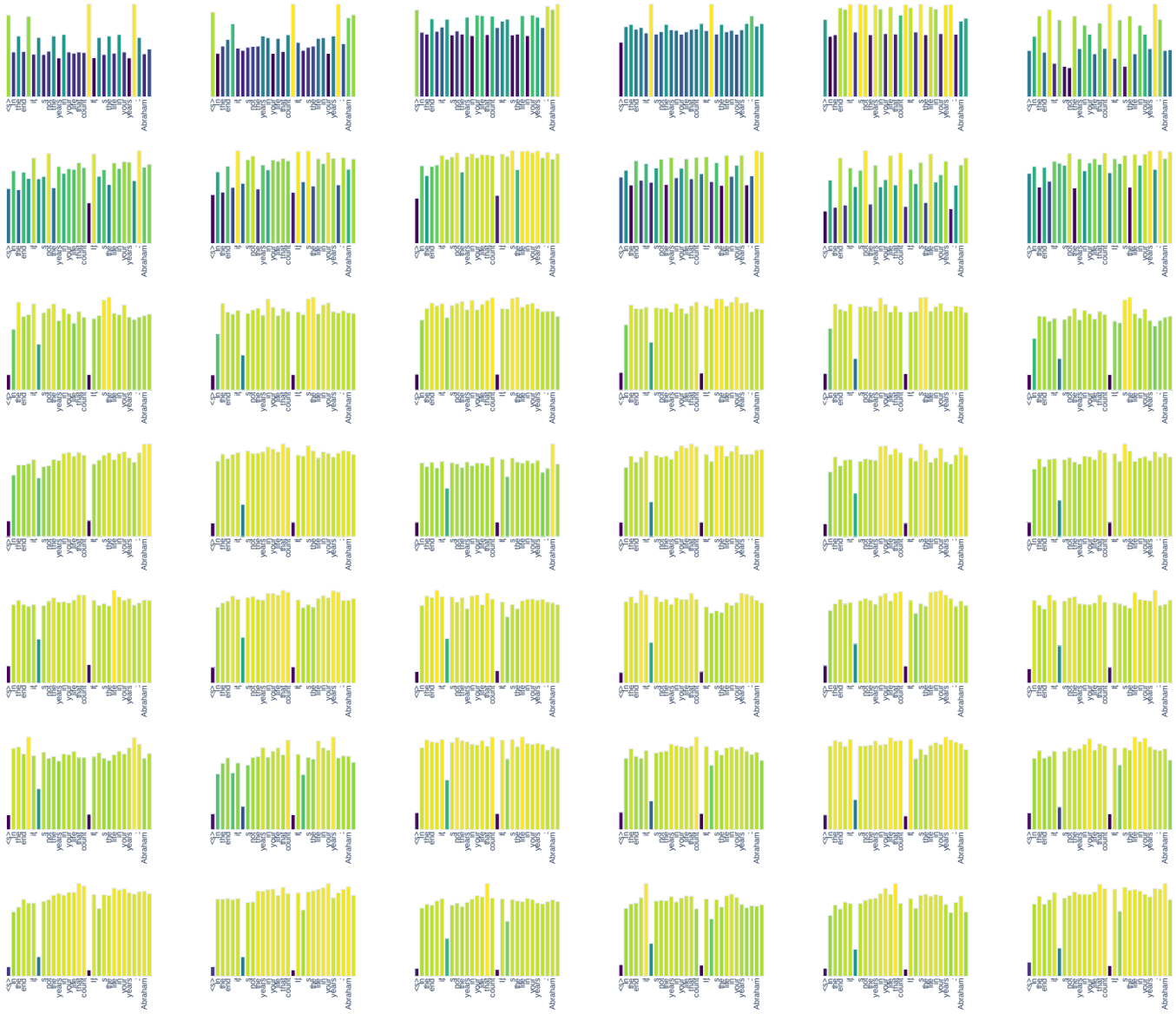


Figure 4.24. Norms of KV cache tokens in Llama2-7B

Figure 4.25. Attention maps in Llama2-7B

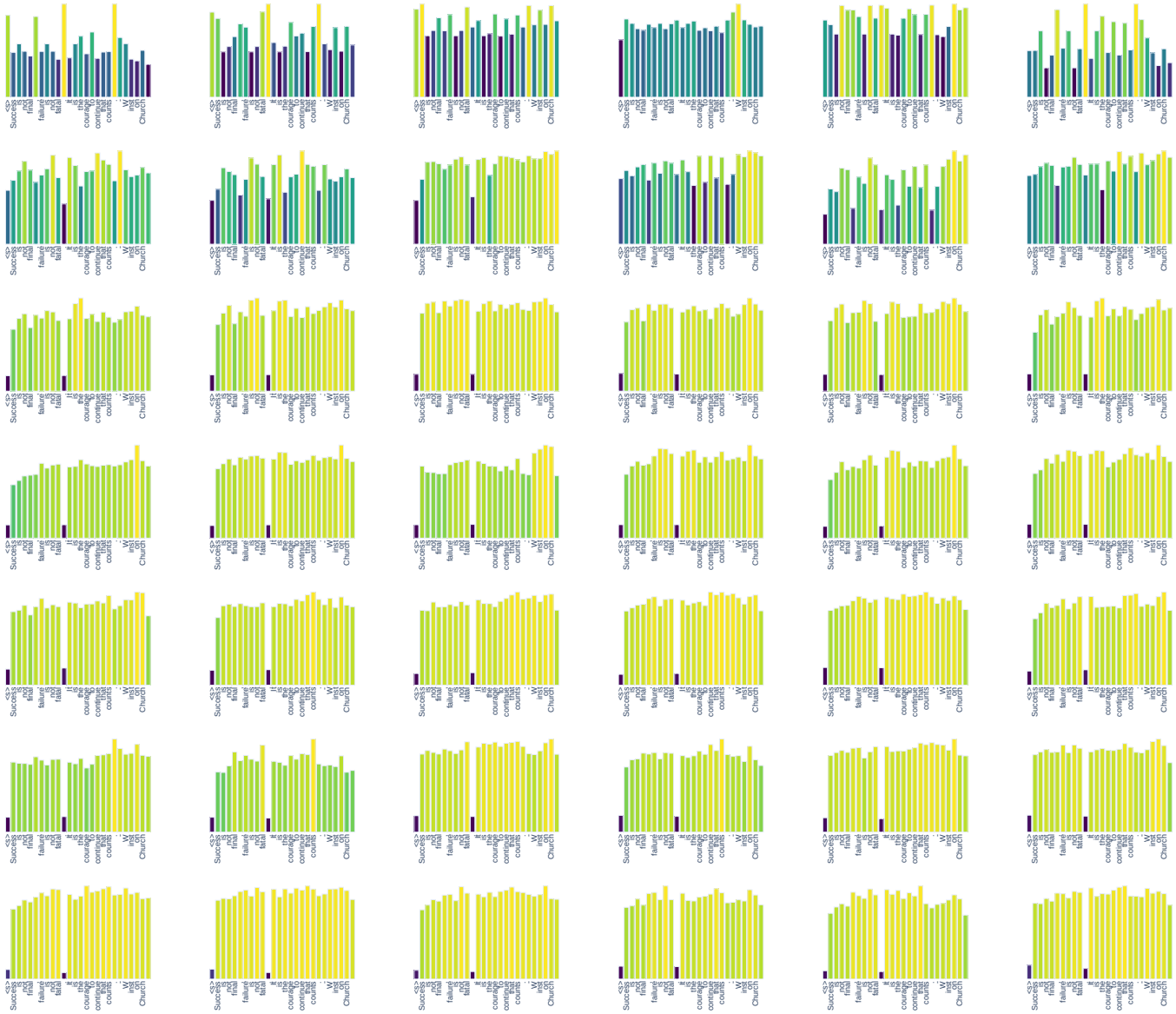


Figure 4.26. Norms of KV cache tokens in Llama2-7B

Figure 4.27. Attention maps in Llama2-7B

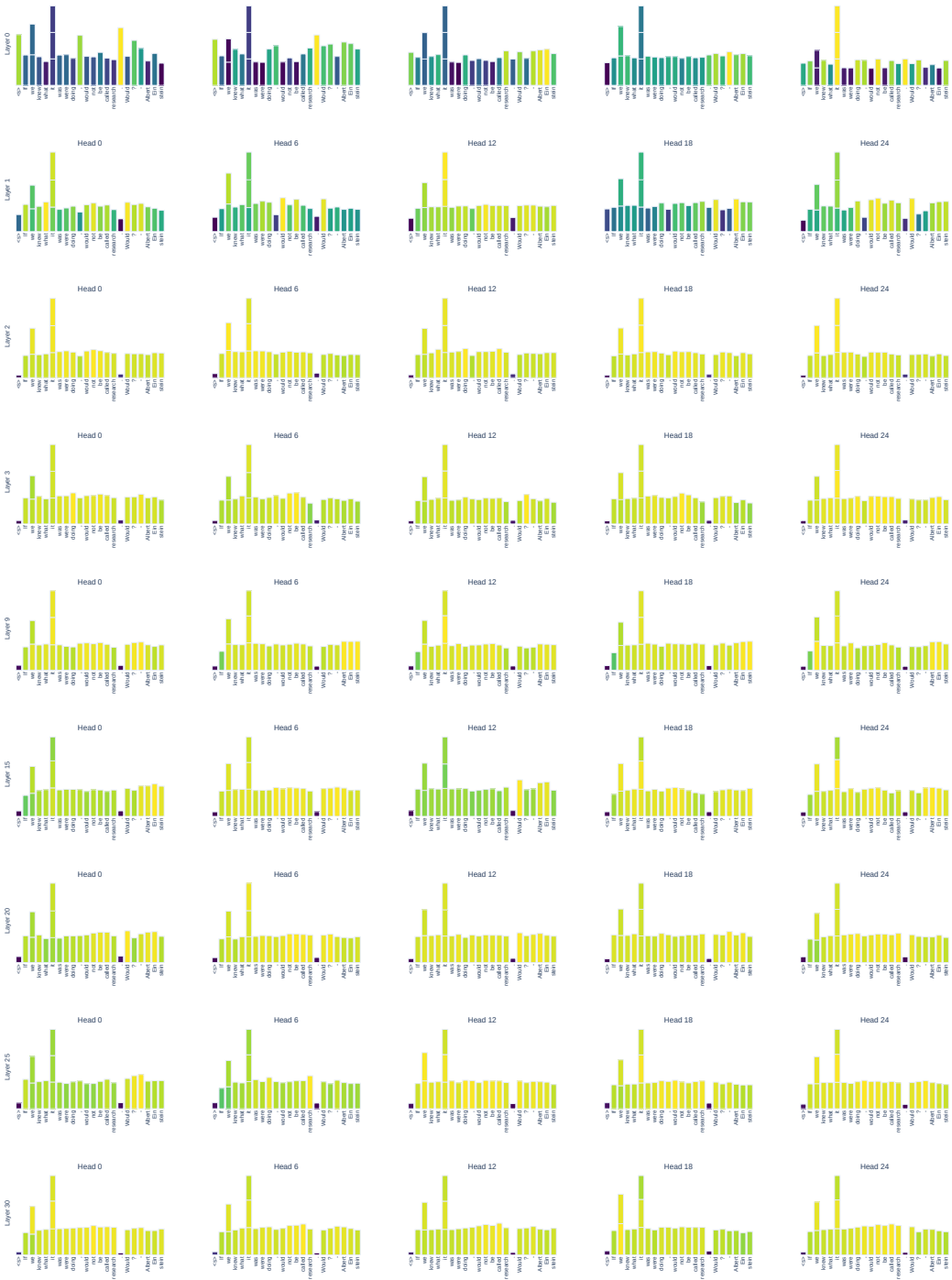


Figure 4.28. Norms of KV cache tokens in Llama2-7B

4.1.11 Additional token embeddings plots

We show in Figure 4.29 some additional figure that represent Llama3-8b token embeddings sparsity.

4.1.12 Experimental setup

In all experiments, we used the HuggingFace library and did not change the model's default hyperparameters. For language modelling, results are averaged across 50 samples. The Fig. 4.7 and Fig. 4.2 are the average results of 1024 examples with a chunk size of 1024 using Wikipedia.

4.1.13 Conclusion

In this paper, we introduced a simple yet highly effective strategy for KV Cache compression in LLMs based on the L_2 norm of key embeddings. We show that there is a significant correlation between the L_2 norm of a key embedding and its attention score. Leveraging this observation, we compress the KV Cache by retaining only those keys with the lowest L_2 norm. Our experimental results on various tasks show that our compression strategy maintains the predictive accuracy of the model while significantly reducing the memory footprint. Our approach is straightforward and can be applied directly to any transformer-based, decoder-only LLM.

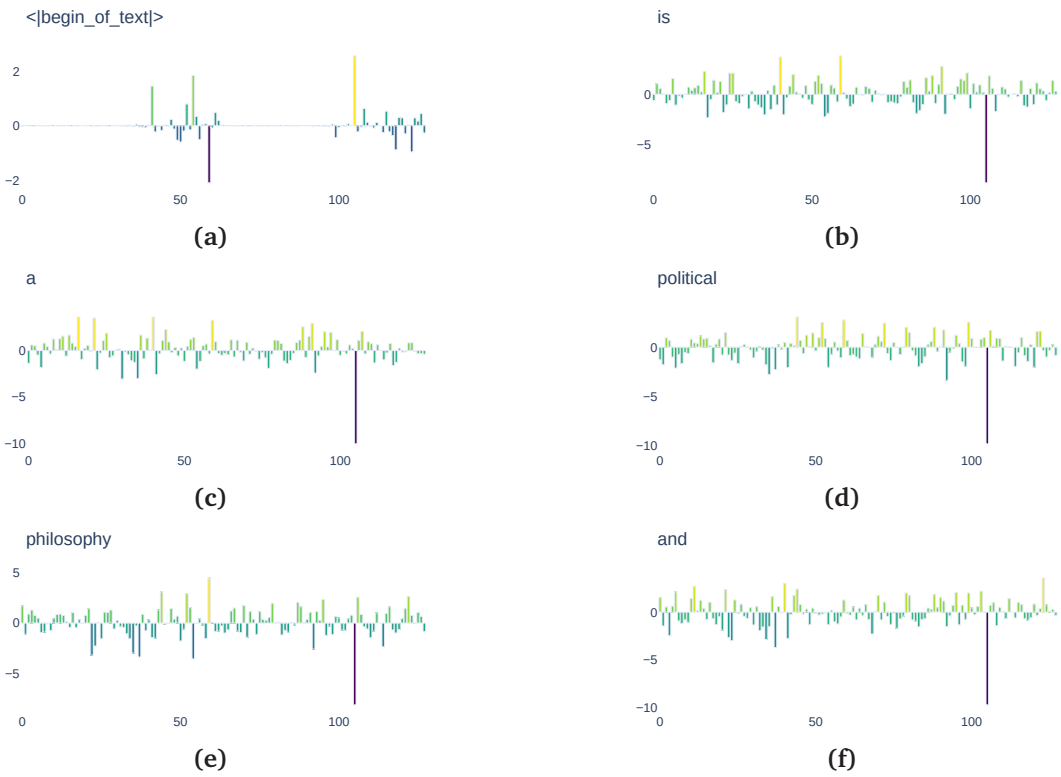
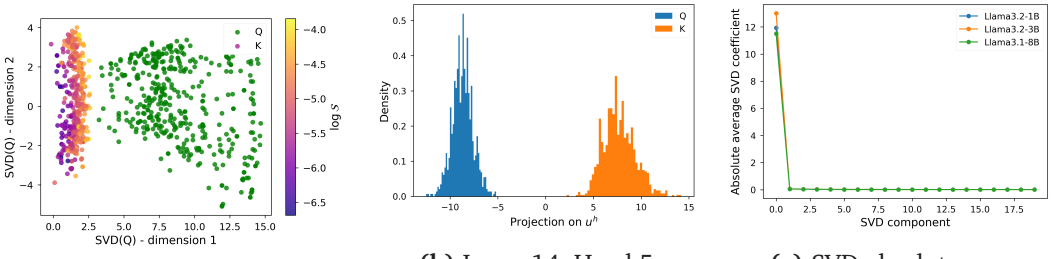


Figure 4.29. Key projections of Llama3-8b of the bos $|beginoftext|$ token vs other tokens. Each value represents the activation in a specific dimension for the embedding of the key projection. We found similar patterns across almost all heads and layers and in multiple texts.

4.2 QFilters: Leveraging Queries and Keys geometry for KV Cache Compression

The performance of Large Language Models (LLMs) as autoregressive text-generation systems relies on the effectiveness of the Transformer architecture [584]. Recently, long-context models such as Gemini-Pro-1.5 [497], Claude-3 [19], GPT-4 [2], and Llama-3.1 [171] have demonstrated the ability to process hundreds of thousands of tokens. However, processing such long sequences comes with significant challenges, as it may lead to higher decoding latency and memory saturation. As the context length grows, each inference step involves storing an increasingly large context from GPU memory in the form of the KV Cache, creating a memory bottleneck that hinders efficient inference [188]. To address this issue, KV Cache compression methods aim



(a) Layer 0, Head 21

(b) Layer 14, Head 5 ($\epsilon = -1$)

(c) SVD absolute avg. coefficients

Figure 4.30. Left and center: distributions of the projections of Q^h and K^h on u^h for Llama-3.1-8B. Right: estimates of $\left|\mathbb{E}_i(\langle Q_i^h, v_m\rangle)\right|$ where v_m are the right vectors from the SVD of a set of Q^h representations from different Llama models, averaged over all layers and heads.

to reduce the size of this past-context representations storage by removing or merging Key-Value pairs, thereby alleviating memory bottlenecks. While KV Cache compression techniques have gained popularity, many approaches require fine-tuning or re-training the underlying models [421, 8, 141], which limits their applicability in real-world deployment scenarios. Training-free methods have also been proposed, but they often rely on access to attention weights to evaluate the importance of Key-Value pairs [623, 348], making them incompatible with the widely adopted efficient attention algorithm FlashAttention [128]. These methods usually require a partial re-computation of the attention matrices, which leads to a time and memory overhead. Hence, these algorithms are often used to compress prompts before generating answers and are not ideally suited for memory-constrained generation.

In this work, we propose Q-Filters, a training-free KV Cache compression method that uses the geometric properties of **Q**ueries and Keys to **filter** out the less important Key-Value pairs. Our approach achieves competitive results across synthetic tasks and pure generation cases while maintaining compatibility with FlashAttention and, thus, better time efficiency.

Analysing the properties of queries (Q) and Keys (K) distributions, we find that *a single direction, spanned by the principal component of Q, encodes an input selection process for each head*. Identifying this direction allows us to efficiently estimate which inputs are mostly ignored by a given head and can thus be discarded with minimal performance loss. Interestingly, we find that this direction is context-agnostic, *i.e.,* the

directions we identify in different contexts are highly consistent. Leveraging this property, we calculate lightweight projections, which we refer to as Q-Filters, based on a small held-out calibration dataset only once for every model, incurring minimal computational overhead. At inference time, we use Q-Filters to project Keys in the pre-computed direction to estimate the importance of Key-Value pairs without accessing attention scores, and we prune the KV Cache accordingly. This makes our method faster than most KV Cache compression alternatives that use attention scores to estimate the importance of the KV pairs.

Additionally, our method is training-free, requiring only a very short initial calibration, and we show it can be easily applied to a variety of decoder-only language models. We validate our method on a wide set of tasks, ranging from language modelling to in-context learning and long-context tasks, achieving competitive performance even with 32x compression ratios.

4.2.1 Method

Exploring the Query-Key Geometry

In [157], the authors examined a relationship between basic characteristics of the Key representations and attention score distributions. Notably, they observe a negative correlation between the average attention weight given to a position and the L_2-norm of the K_t^h vector at that position (where h is the head index, and t the position in the sequence). Leveraging this observation, they propose to compress the KV Cache by selecting the KV pairs for which $||K_t^h||_2$ is the smallest. Using this simple heuristic, they are able to reach $\times 2$ compression ratios without altering the retrieval and modelling performance of the models they study.

[209] show that the distributions of Q_t^h and K_t^h are *anisotropic*, i.e. they do not uniformly occupy $\mathbb{R}^{d_H}$. They observe that both distributions "drift away" from the origin as training progresses. Crucially, this drift occurs along parallel directions in $\mathbb{R}^{d_H}$, so that the dot product between mean Q_t^h and mean K_t^h representations tends to increase in absolute value and to be either positive or negative for different heads.

Motivated by [157] and [209], we propose to further explore some geometrical properties of Q^h and K^h vectors and their implications for unnormalized attention logits $Q^h(K^h)^T$. First, we expose the *joint* anisotropy of Q^h and K^h in Figure 4.30b. Empirically, we observe a head-dependent direction u^h so that projections of Q_t^h and K_t^h in this direction are each of consistent sign. This direction can be identified by performing a Singular Value Decomposition (SVD) on a set of Q^h representations and setting $u^h = \pm v_1$ where v_1 is the right-vector corresponding to the highest singular value, and where the sign is chosen to make most Q_t^h (if not all) project to a positive value.

Interestingly, as shown in Figure 4.30c, v_1 is the only component of the SVD that projects Q^h representations to a non-null average coefficient. The intuitive consequence of these two observations is that if a given K_t^h has a strong (negative) projection along u^h, then the $\langle Q_i^h, K_t^h \rangle$ dot products should *all* be negatively affected, which results in less important attention weights towards K_t^h. Since v_1 is the only direction that encodes such a phenomenon, one can derive an approximation of the average attention score received by K_t^h based on its projection on $v_1 = u^h$, which is summarized in Theorem 4.2.1.

Proposition 4.2.1 (proof in Section 4.2.4)**.** *Under Theorem 4.2.2 and Theorem 4.2.3, we have:*

$$\mathbb{E}_{Q_i^h}(\langle Q_i^h, K_j^h \rangle) \approx \kappa^h \langle K_j^h, u^h \rangle$$

where κ^h is a positive constant.

Intuitively, projecting K_t^h along the anisotropic direction u^h gives us an estimate of the attention logits that involve K_t^h up to a positive multiplicative constant κ^h.

The resulting v_1 vectors, that we refer to as Q-Filters, allow to estimate, up to a sign, which Key-Value pairs are worth storing for each head along generation.

Q-Filters

Based on Theorem 4.2.1, we can design a KV Cache compression scheme that consists of the following steps:

1. For a given model, retrieve its Q-Filters, which can be obtained with the following procedure:
   (a) Gather Q^h representations by passing samples through the model;
   (b) Compute the SVD of the gathered representations at each layer and head;
   (c) Obtain the *positive* right vector (or Q-Filter) for each head $v_1^+ = \text{sgn}(\mathbf{1}u_1^T)v_1$.
2. At inference, for each head, discard the K_t^h with the lowest $\langle K_t^h, v_1^+ \rangle$ value.

In the case of Grouped-Query Attention or GQA [8], we simply average the Q-Filters for each group of Query representations.

We bring the attention of the reader to the fact that this method only requires a single preparation step following training for a given model. The Q-Filters are entirely context-agnostic and rely on inherent properties of the Query and Key latent spaces. In the rest of this article, we use a subset of the Pile dataset [193] to compute the Q-Filters and discuss the choice of the dataset and of the number of necessary SVD samples in Section 4.2.10.

In Figure 4.40, we observe that the Q-Filters heuristic is noticeably more correlated with the attention score for most heads compared to the L_2-norm metric. As such, ordering the Key-Value pairs using the Q-Filters heuristic should allow us to select more relevant pairs than using the method from [157] - that we will call K-norm for the sake of simplicity.

4.2.2 Experiments

We validate our method both on memory-constrained language modelling and on long-context retrieval tasks (e.g. needle-in-a-haystack). Additionally, we test our method on the Ruler dataset [263], which is specifically designed to test the model's long context modelling abilities. We test Q-Filters on Llama-3.1-8B, Llama-3.1-70B [171], and Qwen-2.5-7B [486], but the method can be easily used for any pre-trained decoder-only LLM. We compare Q-Filters with several KV Cache compression methods. These include StreamingLLM [623], which focuses on language modelling by always retaining the initial tokens of the sequence. We also compare with SnapKV [348], which performs

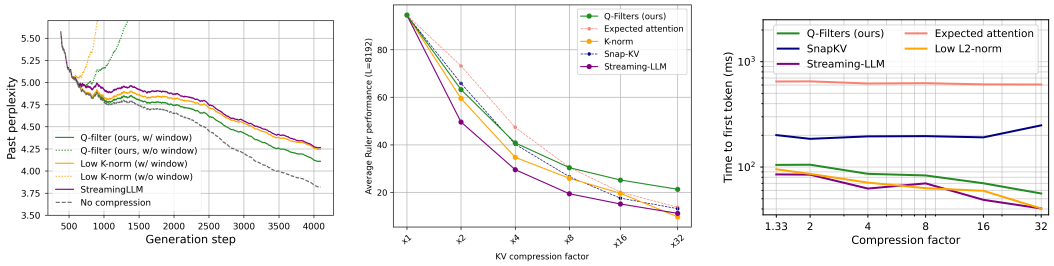


Figure 4.31. Left (a): Llama-3.1-8B Language Modelling performance in the streaming compression setup. Center (b): Average performance on Ruler (8192) for Llama-3.1-8B-Instruct. Right (c): First token latency across KV Cache compression methods of Llama-3.2-8B with a length of 64k prompt.

compression based on attention scores from the final portion of the prompt, making it particularly suitable for compression of large prompts. Additionally, we compare against preserving low-L_2 norm tokens [157] and the recent ExpectedAttention [279].

We show results for different tasks in Figure 4.31. In Language modelling, we observe that Q-Filters consistently achieves the lowest perplexity among compression schemes, even for very long contexts. In the Ruler dataset [263], we test the model's score for several compression factors ranging from 2× to 32×. While for some lower compression factors, we find performance on par with other methods, Q-Filters achieve the highest score with the strongest compression factor of 32×, demonstrating the method's effectiveness at high compression rates. We provide detailed results in table Table 4.1 and further discussion on additional benchmarks in Section 4.2.9.

Compression method	FA-compatible	CWE	FWE	Multi-Key	Multi-Query	Multi-Value	Single	QA	VT	Average
SnapKV		**88.7**	89.0	15.1	29.6	28.8	68.7	42.8	83.2	50.5
Expected Attention		70.0	79.3	12.0	**59.7**	37.8	31.2	44.2	96.3	43.2
Streaming-LLM	✓	53.8	**93.4**	14.1	16.8	16.7	15.7	**62.3**	15.8	31.6
K-Norm	✓	22.9	74.8	8.7	16.6	25.8	55.9	20.6	32.0	31.3
Q-Filters (ours)	✓	82.5	80.2	**22.9**	49.1	**60.6**	**71.1**	37.6	**100**	**56.1**

Table 4.1. Results on the Ruler-4096 dataset for Llama-3.1-70B-Instruct with an 8× compression ratio. The second column indicates compatibility with FlashAttention.

Throughput and Scalability

Our approach is more efficient than many KV Cache compression methods, as it estimates the relevance of a K^h representation *without materializing the attention maps*. This property makes it compatible with memory-efficient self-attention implementations such as FlashAttention [128]. During inference, Q-Filters maintains the same theoretical time complexity as the K-norm method [157], since computing a norm and a scalar product require a comparable number of floating-point operations.

By avoiding the explicit computation of attention scores, our method achieves lower inference latency compared to existing approaches. To quantify this efficiency, we measure the *Time to First Token* across different methods in Section 4.2.2. Time to First

Token (TTFT) refers to the latency between submitting a prompt and receiving the first generated token. This metric is particularly relevant in scenarios where fast response times are critical, such as interactive AI applications. Compressing the KV Cache directly impacts TTFT: by reducing the memory footprint of the KV Cache, it allows a larger portion of the prompt context to fit within fast-access memory, minimizing memory swapping overhead. As a result, compression techniques that efficiently manage the KV Cache should significantly reduce initial response latency. Notably, our experiments show that Q-Filters maintain this performance advantage even as the sequence length increases, suggesting better scalability compared to methods that require explicit attention computation.

4.2.3 Observations

Hypothesis 4.2.2 (Joint anisotropy)**.** *There exist* $u^h \in \mathbb{S}^{d_H-1}$ *and* $\epsilon = \pm 1$ *such that*

$$\mathbb{E}\left(\langle Q_i^h, u^h\rangle\right) > 0 \quad \textit{and} \quad \mathbb{E}\left(\langle K_j^h, \epsilon u^h\rangle\right) > 0,$$

where $\langle\cdot,\cdot\rangle$ *denotes the dot product.*

To validate Theorem 4.2.2, we compute the Singular Value Decomposition (SVD) of a set of Q^h representations taken from various sequences for Llama-3.1-8B. We find that the first right-vector of the SVD verifies Theorem 4.2.2 for all tested heads, and we display examples of projection distributions in Figure 4.30b. The intuitive consequence of this observation regarding attention weights is that, if a given K_t^h has a strong projection along ϵu_h, then future queries $Q_{\geq t}^h$ can be expected to have a stronger dot-product with K_t^h in average.

However, it is not clear *a priori* that this effect is uni-directional, i.e. that there exists a unique direction u^h (up to a sign) that verifies Theorem 4.2.2. Hence, identifying one such direction may not suffice to characterize the anisotropy of Q^h representations and to derive estimations of the dot-products used in attention. The uni-directional nature of the Query-Key anisotropy can be formalized as in Theorem 4.2.3.

Hypothesis 4.2.3. *Let* $u^h = _{u\in\mathbb{S}^{d_H-1}} \mathbb{E}\left(\langle Q_i^h, u\rangle\right)$ *and* $B = (u^h, u_2, ..., u_{d_H})$ *an orthonormal basis of* $\mathbb{R}^{d_H}$*. Then for all attention inputs* X*:*

$$\forall m \in [2, d_H], \mathbb{E}\left(\langle Q_i^h, u_m\rangle\right) \approx 0$$

In Figure 4.30c, we observe that only the first singular component of the SVD of Q^h representations carries an anisotropic behavior, as the projections on all other components have a null mean. Hence, by taking the SVD right-vector basis as B, we can show that the first component of the SVD empirically verifies Theorem 4.2.3.

4.2.4 Proof of Theorem 4.2.1

We begin the proof by writing $\langle Q_i^h, K_j^h\rangle$ in the basis B:

$$\mathbb{E}_{Q_i^h}(\langle Q_i^h, k\rangle) = \mathbb{E}_{Q_i^h}(\langle Q_i^h, u^h\rangle)\langle k, u^h\rangle + \sum_{m=2}^{d_h} \mathbb{E}_{Q_i^h}(\langle Q_i^h, u_m\rangle)\langle k, u_m\rangle$$

Theorem 4.2.3 states that $\mathbb{E}_{i,X}(\langle Q_i^h, u_m \rangle) \approx 0$, which lets us do the following approximation:

$$\sum_{m=2}^{d_h} \mathbb{E}_{Q_i^h}(\langle Q_i^h, u_m \rangle)\langle k, u_m \rangle \approx 0$$

By combining Theorem 4.2.2 and Theorem 4.2.3, we also have that:

$$\mathbb{E}_{Q_i^h}(\langle Q_i^h, u^h \rangle) > 0$$

We conclude the proof by setting $\kappa^h = \mathbb{E}_{Q_i^h}(\langle Q_i^h, u^h \rangle)$.

Remark This result provides a justification for the method developed in [157]. As a matter of fact, Theorem 4.2.2 implies that $\mathbb{E}_j\left(\cos(K_j^h, u^h)\right)$ should have the same sign as ϵ. In practice, we observe $\epsilon = -1$ for a vast majority of heads in trained causal LMs. Hence, we can derive a looser estimation from Theorem 4.2.1:

$$\mathbb{E}_{i,X}(\langle Q_i^h, K_j^h \rangle) \approx -\kappa^h \left| \mathbb{E}_{j,X}\left(\cos(K_j^h, u^h)\right) \right| ||K_j^h||_2$$

This estimation shows that the L_2-norm of K_j^h vectors is negatively correlated with the corresponding mean attention logits and can therefore be used to approximate them. However, only using the L_2-norm to estimate the attention score as done in [157] is suboptimal, as it ignores the angular component of the $\langle K_j^h, u^h \rangle$ product.

4.2.5 Generation Results

We compute the final perplexity of Llama-3.1-70B in the memory-constrained setup for various compression factors and methods.

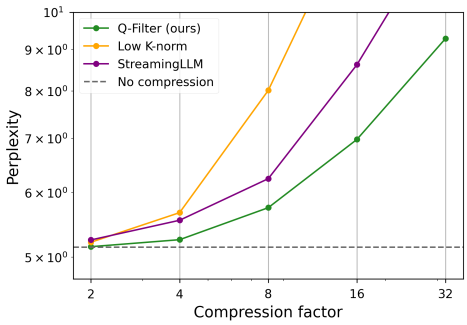
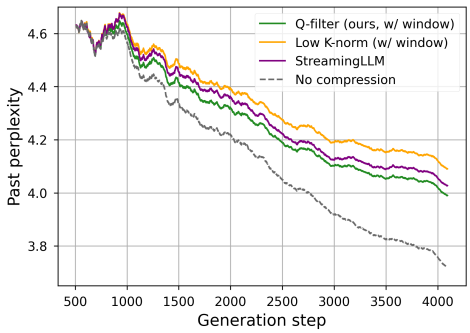
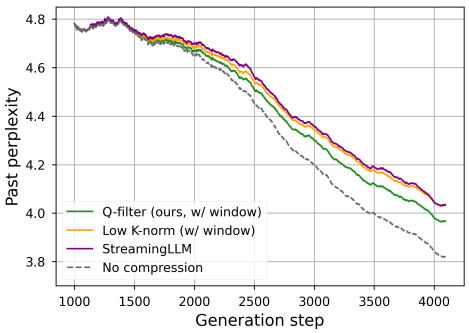


Figure 4.32. Final perplexity after 512 tokens for Llama-3.1-70B in the memory-constrained generation scenario.

We also run a study similar to the one conducted in Figure 4.35 with Qwen-2.5-7B-Instruct, which we display in Figure 4.33a, and with Llama-3.2-1B, which we display in Figure 4.33b.



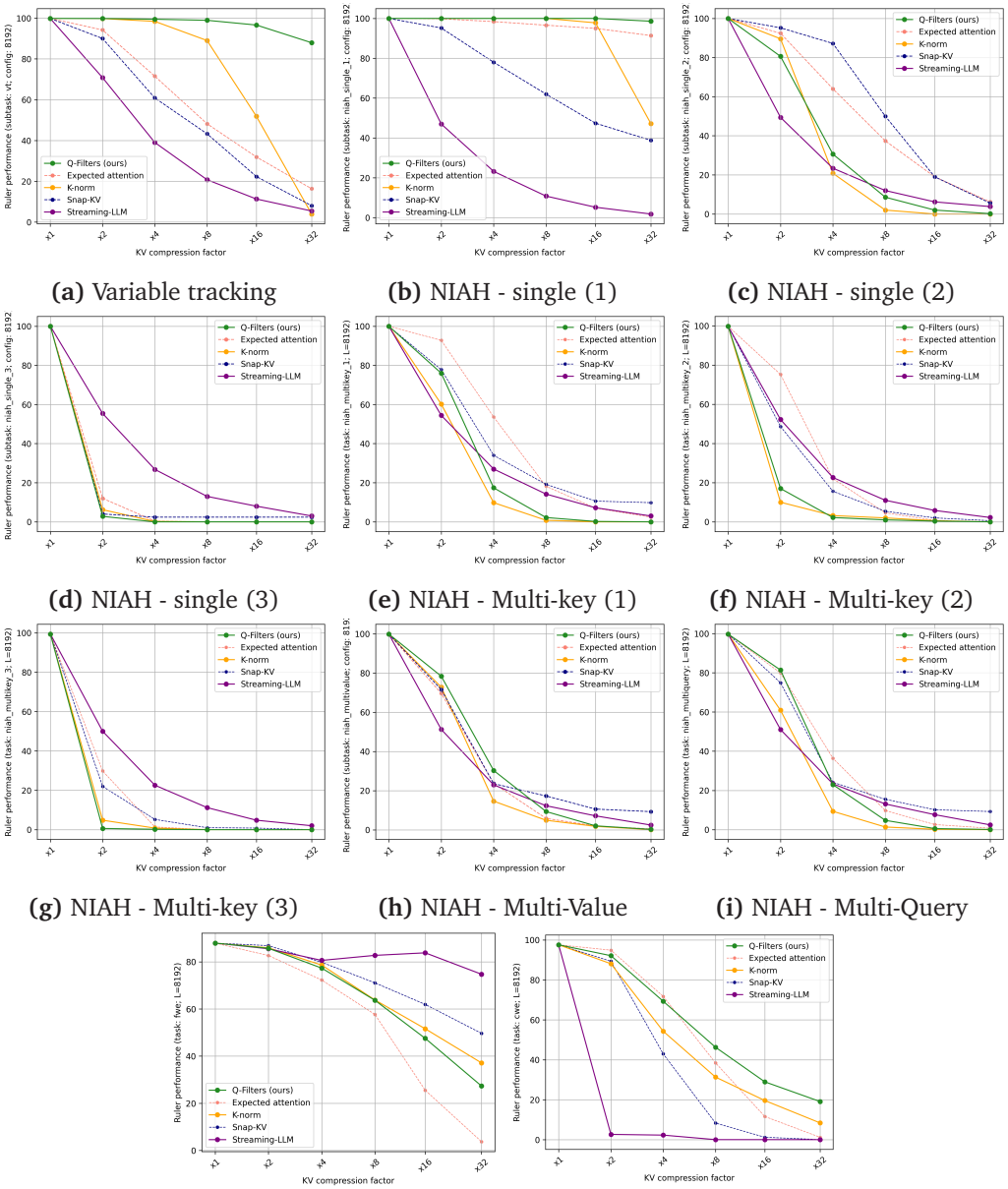
(a) Perplexity of the Qwen-2.5-7B-Instruct model along generation.



(b) Perplexity of the Llama-3.2-1B model along generation.

4.2.6 Ruler Results

In Figure 4.34 we report detailed evaluation on the subsets of the Ruler dataset [263].



(a) Variable tracking **(b)** NIAH - single (1) **(c)** NIAH - single (2)

(d) NIAH - single (3) **(e)** NIAH - Multi-key (1) **(f)** NIAH - Multi-key (2)

(g) NIAH - Multi-key (3) **(h)** NIAH - Multi-Value **(i)** NIAH - Multi-Query

(j) Frequent Words Extraction (FWE) **(k)** Common Words Extraction (CWE)

Figure 4.34. Performance of Llama-3.1-8B-Instruct using several KV Cache compression methods on individual tasks from the Ruler dataset (with length 8192) as compression ratio evolves. We report prompt compression methods using dotted lines for comparison.

4.2.7 Generation examples

Using Llama-3.1-8B, we identify interesting cases where Q-Filters provide the correct next token in a given long context, while K-norm and Streaming-LLM fail to capture the relevant information.

Text Sample (Context)	Q-Filters	K-Norm	Streaming-LLM
One of the show's first longest-running storylines was the rivalry between a young manicurist Jill Foster Abbott (Brenda Dickson, Jess Walton) and wealthy socialite, ***Katherine*** *Chancellor (****Jeanne*** *Cooper). [...] After much investigation, it is revealed that Kay is Jill's biological...*	*mother*	*father*	*father*
Both extreme right-wing leaders taught and practised the theology of ***Christian Identity****, a belief system which the FBI includes on its watch list as an extremist religion. [...] Here, the group trained an estimated 1,500 of like-minded Christian...*	*Identity*	*fundamental*	*fundamental*
The Viral Fever *[...]* ***TVF*** *debuted their platform, releasing the final two episodes of Pitchers on* ***TVFPlay****. [...]* ***TVF*** *claims to have worked with over 150 brands. [...] The show has been on hold as writer Biswapati Sarkar focuses on writing web series, including the sequel to TV...*	*F*	*_show*	*_show*

Table 4.2. Next-token generation examples for different KV Cache Compression methods, applied to Wikipedia article. Passages in bold correspond to useful information that is necessary to resolve the ambiguity in the choice of the next token.

4.2.8 Implementation Details

For all our experiments, we use the popular Huggingface models with the recently released KVPress library [279].

4.2.9 Detailed Experimental Results

Language Modelling To evaluate the performance of Q-Filters in the language modelling setup, we perform generation on the Pile dataset [193]. We let the KV Cache grow up until a certain threshold, after which we start evicting the KV pairs so that the total size never exceeds the maximum threshold. We measure performance by tracking the model perplexity computed on past tokens in 20 sequences. We report results for a maximum KV Cache size of 512 pairs in Figure 4.35. We observe that Q-Filters consistently achieves the lowest perplexity among compression schemes, even for very long contexts. This observation scales to the 70B model, where Q-Filters significantly

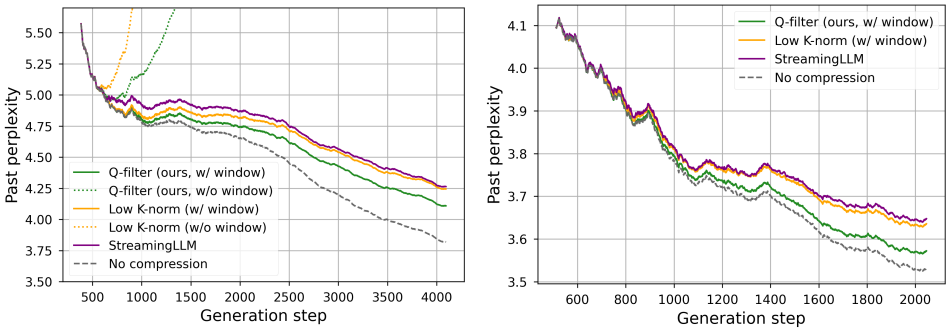
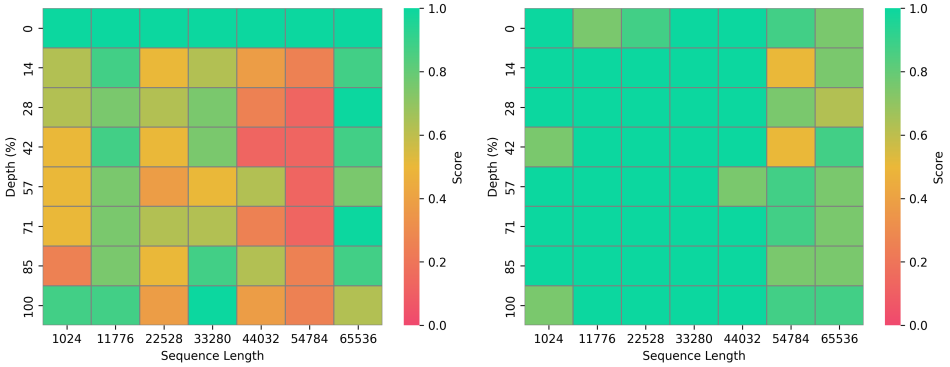


Figure 4.35. Generation performance for a KV Cache size limited to 512 items for Llama-3.1-8B (left) and Llama-3.1-70B (right).



(a) K-norm (average accuracy: 63%) **(b)** Q-filters (average accuracy: 91%)

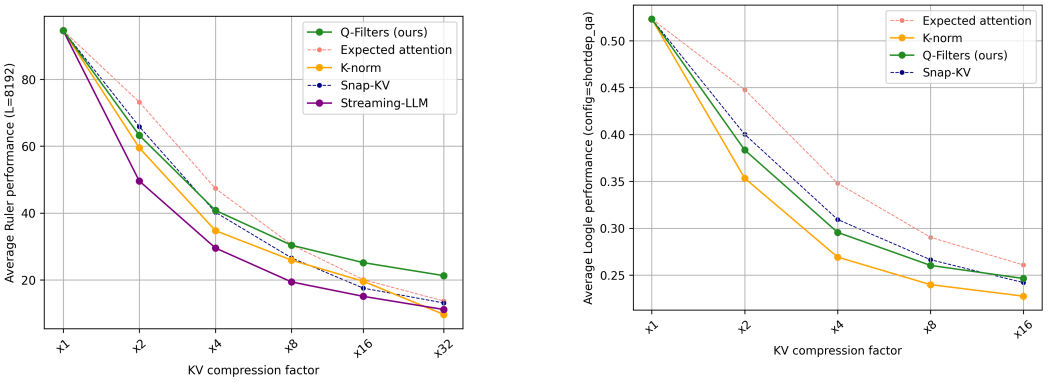
Figure 4.36. Needle-in-a-haystack performance for Llama-3.1-8B using 64x KV Cache compression.

reduces the perplexity gap. This improvement is more pronounced in the latter portions of the sequences, suggesting better retention of relevant contextual information.

Needle in a Haystack The Needle-in-a-Haystack task embeds a key piece of information (the "needle") within a long sequence of distractors (the "haystack"), followed by a question that requires retrieving the needle. This evaluates the model's ability to handle long-range dependencies and tests how well KV Cache compression retains critical information. If important KV pairs are evicted, the model fails to answer correctly.

We evaluate Q-Filters by placing the needle at depths from 1k to 64k tokens and measuring retrieval accuracy. As shown in Figure 4.36, Q-Filters outperforms K-Norm [157], preserving crucial information even in extremely long contexts.

Ruler Tasks We evaluate the proposed method on the Ruler dataset [263], which comprises several sub-tasks that test the model long context modelling abilities, including Multi-hop Tracing, Long Context Aggregation, Long Context Retrieval and Question Answering. The dataset offers 3 variants with different sequence lengths: 4096, 8192, and 16384. We compare the score on Ruler with several other KV Cache compression



(a) Average performance on Ruler (8192)

(b) Average performance on Loogle (Short Dependency QA)

Figure 4.37. Average score for different long-context benchmarks using Llama-3.1-8b with different methods and compression ratios

methods and show average results in Figure 4.37a. We report detailed per-task results in Table 4.1 and in Section 4.2.6. We test the model's score for several compression factors ranging from 2× to 32×. While for some lower compression factors, we find performance on par with other methods, Q-Filters achieve the highest score with the strongest compression factor of 32×, demonstrating the method's effectiveness at high compression rates.

4.2.10 Robustness of the Calibration Dataset

In Figure 4.38, we analyse how the calibration dataset size impacts the performance of our Q-Filters computation. Our experimental results demonstrate that increasing the number of samples in the calibration dataset leads to an improvement in average perplexity, although the marginal benefits diminish beyond a certain point, namely around 1k samples. This suggests that while larger calibration datasets generally produce more robust Q-Filters, there exists a practical trade-off balancing computational cost and performance benefits. Based on these empirical findings and computational efficiency considerations, we standardized our experimental protocol to utilize 3,000 samples for computing the Q-Filters across all subsequent experiments. Another important consideration in the development of robust Q-Filters is the choice of calibration dataset. To investigate this aspect, we conducted a systematic analysis using multiple diverse datasets and model versions in Figure 4.39. Our experiments revealed that the Q-Filter vectors exhibit remarkable stability across different calibration datasets, with a high average cosine similarity between vectors computed from distinct sources. This finding suggests that our method is relatively insensitive to the specific choice of calibration data, provided it maintains sufficient diversity and quality. Based on these results, we opted to use a carefully curated subset of the Pile dataset [193] for all Q-Filter computations.

4.2.11 Q-Filters Estimation Overhead

It could be argued that our method introduces a memory overhead as we need to store the Q-Filters on-device. Nevertheless, for a model using l layers and n_H heads, storing

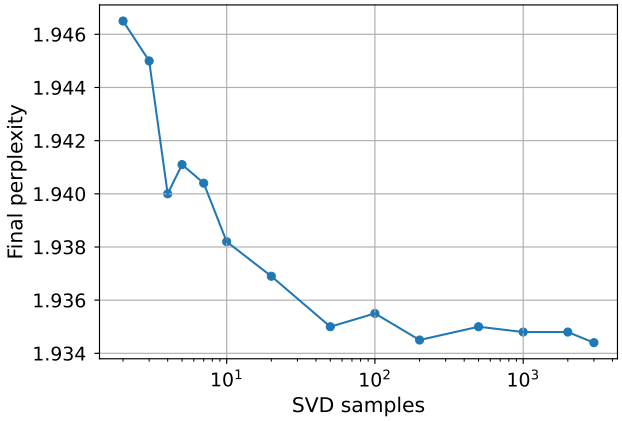


Figure 4.38. Perplexity after 1024 tokens for Q-Filters obtained using different counts of Q^h representations to calculate the SVD.

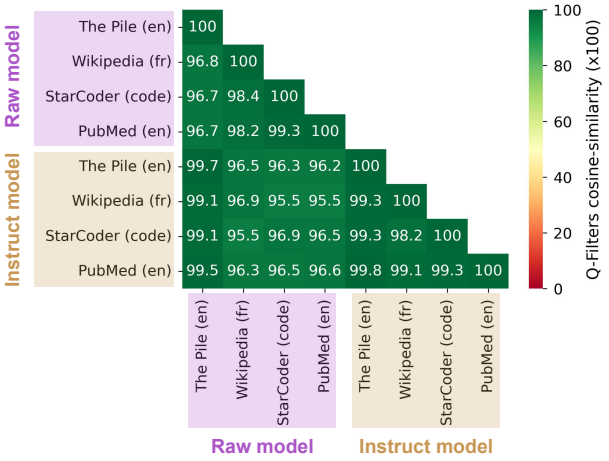
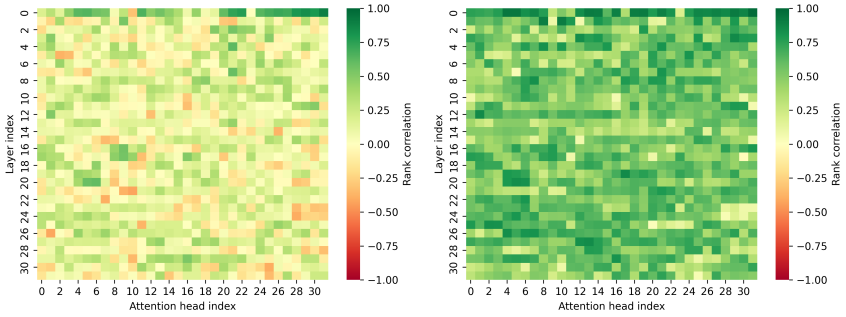


Figure 4.39. Cosine-similarity between Q-Filters computed on datasets coming from different domains and languages and on pre-trained and post-trained models. The scores are averaged over all layers and heads.

the Q-Filters requires $l \times n_H \times d_H$ parameters. For Llama-3.2-1B, this is 36k× smaller than the total parameter count and 196k× smaller in the case of Llama-3.2-405B. Another source of overhead could be attributed to the initial computation of the filters that are required for every new model. We find that passing 20 samples of length 2048 through the model and performing the SVD on 3k randomly sampled representations for each head is sufficient to obtain strong performance. In our experiments with Llama-3.2-70B, computing the filters took less than 3 minutes on two A100-80GB GPUs. This cost is thus negligible when compared with the cost of inference.

4.2.12 Related Work

After the success of long-context models [497, 19, 2], compressing the KV Cache has become a key research focus to enable processing of long-context inputs.



(a) K-norm **(b)** Q-Filter

Figure 4.40. Spearman rank correlation between KV compression scoring metrics and the observed attention S^h for Llama-3.2-1B, for K-norm (top) and Q-Filters (bottom).

Some methods reduce the KV Cache size by modifying the model architecture. For example, [8] and [529] reuse the same Keys for multiple queries, thereby reducing redundancy in storage. [421] propose a dynamic token-merging strategy, learning which KV pairs to merge. While these approaches achieve significant compression, they require training or fine-tuning, making them less practical in real-world scenarios where retraining the model from scratch is not feasible. In contrast, our method requires only a short, computationally inexpensive calibration step, avoiding parameter updates entirely. Recently [141] introduced a Multi-Head Latent Attention, a modification to the standard attention mechanism that performs a low-rank reduction of the KV Cache during pre-training.

Training-free approaches aim to compress the KV Cache without modifying the model, typically by approximating the attention score over long sequences and prioritizing tokens with higher importance. Among these, [623] focus on language modelling tasks and propose always retaining the first token(s) (as an attention sink) and the last n tokens in a sliding window. Also, [676] focuses on generation tasks and introduces a policy that evicts tokens during generation based on a scoring function derived from cumulative attention. In contrast, other works focus on the task of compressing a large prompt provided by the user. [348] uses attention from the last part of the prompt to estimate KV pairs importance. With the same goal, [79] assigns more cache budget to lower layers and less to higher layers. Finally, [233] proposes to rescale the KV score of other methods by the L_1 norm of the Values.

In contrast, our approach is not tailored to a specific use case but provides competitive performance across both synthetic tasks and real-world scenarios, including in-context learning and chat-based interactions. Additionally, many of these approaches are incompatible with FlashAttention [128] due to their reliance on accessing the full attention weights, which FlashAttention does not expose.

4.2.13 Limitations

In Section 4.2.5, we run generation experiments on Qwen-2.5-7B-Instruct [486], and we observe that, although the results still favour the Q-Filters method, the gap is less clear compared to the Llama models. Our main hypothesis for this discrepancy lies in the slightly different attention mechanism used in Qwen-2.5 suite, which adds a bias to the

QKV projection. Hence, it is likely that the geometrical observations made in Section 4.2.1 are not accurate in that case. Similarly, initial experiments with Olmo-2 models [434] were unsuccessful, which can be explained by their use of the QK-normalization technique [142]. These different tricks would most likely require an adaptation of our analysis to yield a better approximation of the attention distributions.

4.2.14 Conclusion

We introduced Q-Filters, a novel training-free method for KV Cache compression. We show that projecting the Key representations on the main SVD component of the Query vectors results in an accurate approximation of the attention scores. Q-Filters is extremely efficient and is compatible with FlashAttention as it does not require accessing the attention scores. We validated our method on several tasks (Language modelling, NIAH, Ruler) and models up to 70B parameters, and showed competitive performance with respect to more costly state-of-the-art KV Cache compression methods.

4.3 Expected Attention: KV Cache Compression by Estimating Attention from Future Queries

Large language models (LLMs) [3, 18, 398, 640] have revolutionized text generation and reasoning, enabling advanced applications such as long multi-round dialogues, extensive multimodal intelligence [640, 604], and agentic workflows that ingest massive amounts of data [437, 459, 639]. These applications often require processing extensive contextual information. For example, processing a large codebase or a short video can easily involve analyzing hundreds of thousands of tokens. A critical issue in deploying LLMs in such scenarios is the prohibitive memory consumption of the Key-Value (KV) cache [188, 534, 337].

During autoregressive generation, the KV Cache stores key and value vectors for every processed token, enabling efficient attention computation. However, its memory footprint grows linearly with sequence length, quickly becoming the primary bottleneck for long-context inference. A medium-sized 70B model [399] requires approximately 320 GB of GPU memory for a one-million-token KV Cache, far exceeding most GPU capacities. This challenge intensifies with emerging applications where advanced reasoning models generate thousands of intermediate tokens [139, 640] and agentic systems load massive datasets [438, 459]. While current LLMs promise extended context lengths up to a million tokens [198, 398], hardware constraints saturate GPU memory well before reaching theoretical limits.

State Space Models offer a solution by reducing memory costs [226, 225], yet their inferior performance compared to transformers, especially on long context tasks, limits adoption [280, 397]. Other architectural changes limited to the attention mechanism, such as multi-head latent attention [138] or sliding window attention [282, 199], reduce KV Cache size but do not remove the attention bottleneck and are orthogonal to KV Cache compression methods. Additionally, such methods need to be implemented at training time, limiting their application to pre-trained modern LLMs. This creates demand for training-free KV Cache compression methods that preserve transformer architectures while mitigating memory growth.

KV Cache compression exploits semantic redundancy in natural language: not all tokens equally influence future predictions, and many provide negligible information once their contextual role is fulfilled. This property allows to compress the KV Cache by removing some of the key and values stored in it. However, determining which tokens can be safely removed is far from trivial, as any Key-Value (KV) pair's importance depends on how *future queries* will attend to it. Existing approaches use heuristics like discarding oldest tokens [254, 622] or leverage attention scores from past queries [676, 349, 440], but these strategies are limited for real-world scenarios, and often require accessing attention scores which are not materialized in modern transformer implementations [131].

Instead of relying on heuristics or local attention metrics, we argue that a KV pair's significance is best measured by its global effect on the transformer's output. We quantify this effect by isolating each KV pair's contribution within the residual stream, capturing its influence on the model output. This raises the challenge of estimating *how future queries will attend to each token in the context*, which requires accessing attention scores from the past and from future tokens, that are not available at the time of compression. To address this, we introduce *Expected Attention*, which estimates future attention

allocation leveraging the distribution of future queries. Expected Attention estimates the importance that each token in the context has for queries that have not been generated and accordingly prunes the KV Cache up to 60% while preserving performance quality, requiring no architectural modifications or additional training. We release our code as a comprehensive library benchmarking over 20 state-of-the-art compression methods.

To summarize, our contributions are the following:

- We analyse the distributional properties of LLM activations through the lenses of KV Cache compression and introduce the concept of *Expected Attention* to estimate the importance that current tokens will have in the future.
- We introduce a KV Cache compression method that leverages Expected Attention and evicts irrelevant KV pairs for efficient inference.
- We release all our code as a library, designed for researchers, that allows to easily implement, test and benchmark KV Cache compression methods.

4.3.1 Expected Attention

Key-Value Cache in Autoregressive Transformers

We consider decoder-only language models based on the transformer architecture [586], representing the vast majority of modern LLMs. When an input sequence of tokens $\mathbf{x} = [x_1, x_2, \ldots, x_t]$ is fed to the model, each token x_i is transformed into a hidden state representation $h_i \in \mathbb{R}^h$ and processed by a stack of transformer layers, including feed forward networks and multi-head attention blocks. For brevity and clarity, we focus our analysis on a single layer and attention head, noting that the following analysis naturally extends to multi-head attention, grouped query attention (GQA, **(author?)** 9) and all their variants.

Let $h_i \in \mathbb{R}^h$ denote the hidden state at position i in the sequence. In the attention block, the corresponding Query, Key and Value projections are computed as:

$$q_i = R_i W_Q h_i, \quad k_i = R_i W_K h_i, \quad v_i = W_V h_i \tag{4.8}$$

where d is the attention head dimension, $R_i \in \mathbb{R}^{d \times d}$ is the Rotary Position Embedding (RoPE, **(author?)** 553) matrix at position i, and $W_Q, W_K, W_V \in \mathbb{R}^{h \times d}$ are respectively the learnable projection matrices for query, key, and value in $\mathbb{R}^d$. During autoregressive

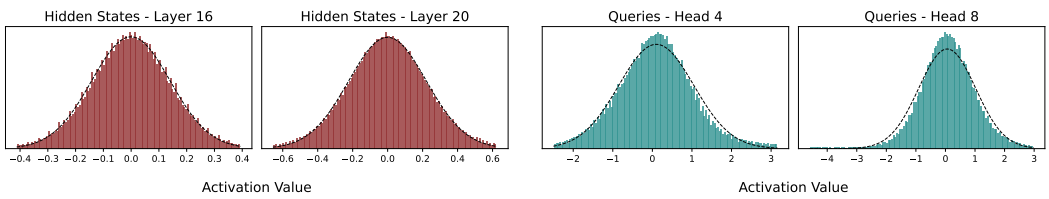


Figure 4.41. Hidden states from layer 16 and 20 and corresponding queries for layer 20 in Llama3.1-8B. Hidden states in modern LLMs are mostly normally distributed. As a consequence, query activations also follow a Normal. The best Gaussian fit is overlayed. We show more examples and discuss this property in Section 4.3.7.

inference, keys and values vectors are stored in the KV Cache to avoid recomputing them in future generation steps. The resulting KV Cache is a collection of Key-Value pairs (k_i, v_i) from all inference steps in the sequence, leading to significant computational savings but increasing memory requirements, growing linearly with sequence length.

At generation step t, the attention mechanism computes the attention score between the current query q_t and each previously cached key k_i for $i \leq t$:

$$a_{ti} = \frac{\exp\left(\frac{q_t^T k_i}{\sqrt{d}}\right)}{\sum_{j=1}^{t} \exp\left(\frac{q_t^T k_j}{\sqrt{d}}\right)} = \frac{z_{ti}}{\sum_{j=1}^{t} z_{tj}} \tag{4.9}$$

where a_{ti} is the normalized attention score between query at position t and key at position i, and $z_{ti} = \exp\left(\frac{q_t^T k_i}{\sqrt{d}}\right)$ represents the unnormalized attention score.

The attention score is used to weight and sum over all values previously stored in the KV Cache. The resulting output is then added to the hidden state h_t:

$$h_t^{\text{out}} = h_t + \sum_{i=1}^{t} a_{ti} W_o v_i = h_t + \sum_{i=1}^{t} \Delta h_{ti} \tag{4.10}$$

where $h_t \in \mathbb{R}^h$ and $h_t^{\text{out}} \in \mathbb{R}^h$ represent the hidden state before and after the attention update respectively, and $W_o \in \mathbb{R}^{d \times h}$ is the learnable output projection matrix. The hidden states embedding h_t represents the "residual stream," [175] updated via vector additions by each transformer block. The value $\Delta h_{ti} = a_{ti} W_o v_i$ isolates the specific residual addition of the i-th KV pair at step t. This decomposition reveals that each cached KV pair (k_i, v_i) contributes a residual update Δh_{ti} to the final output, and provides a natural measure of the importance of each KV pair:

$$\|\Delta h_{ti}\| = a_{ti} \|W_o v_i\| \tag{4.11}$$

where $\|\cdot\|$ denotes the L2 norm. This metric captures both the attention weight a_{ti} (how much the query attends to the i-th key) and the transformed value magnitude $\|W_o v_i\|$ (the impact of the i-th value on the output). Equation 4.11 provides the optimal measure for estimating the importance of each KV pair in the model output. If we could compute this score for all cached KV pairs, we could selectively prune the cache by removing pairs with the lowest impact on the residual stream, thereby minimizing performance degradation. However, computing Equation 4.11 presents significant practical challenges. While $\|W_o v_i\|$ is readily available at inference time, the attention weight a_{ti} depends on future queries that have not yet been generated. Specifically, we cannot know the attention scores from future tokens $t+1, t+2, \ldots$ before computing them, making it impossible to predict which KV pairs will be important for upcoming generation steps. Furthermore, modern transformer implementations utilize Flash Attention [131, 129], which computes attention scores on-the-fly without materializing the complete attention matrix, preventing access to even past attention scores. To address these fundamental limitations, we leverage the properties of activations in modern LLMs, and introduce *Expected Attention*.

Expected Attention: Estimating Attention From Future Queries

Distributional properties of LLM activations To approximate the unnormalized attention score z_{ij}, we leverage the findings of [364], showing that hidden states in modern LLMs loosely follow a Gaussian distribution $h \sim \mathcal{N}(\mu, \Sigma)$. While we show an example of this property in Figure 4.41, we also extensively validate it across multiple model architectures in Section 4.3.7. Given this distributional assumption, queries also inherit Gaussian properties through the linear transformation in Equation 4.8 $q_t = R_t W_Q h_t$:

$$q_t \sim \mathcal{N}(\mu_{q_t}, \Sigma_{q_t}), \quad \text{where } \mu_{q_t} = R_t W_Q \mu, \quad \Sigma_{q_t} = R_t W_Q \Sigma W_Q^T R_t^T \tag{4.12}$$

where $\mu \in \mathbb{R}^d$ and $\Sigma \in \mathbb{R}^{d \times d}$ are the mean and covariance of the hidden state distribution, and $R_t \in \mathbb{R}^{d \times d}$ is the RoPE matrix at position t.

To create a single, tractable representation of attention over a future interval, we approximate the positional embeddings by averaging the RoPE matrix over the next T positions. This gives us a position-averaged query distribution:

$$\bar{q} \sim \mathcal{N}(\bar{\mu}_q, \bar{\Sigma}_q), \quad \text{where } \bar{\mu}_q = \bar{R} W_Q \mu, \quad \bar{\Sigma}_q = \bar{R} W_Q \Sigma W_Q^T \bar{R}^T \tag{4.13}$$

where $\bar{R} = \frac{1}{T} \sum_{j=1}^{T} R_{t+j}$ represents the averaged RoPE matrix over T future positions.

```python
def compress(queries, keys, values, compression_ratio):
    # Compute query statistics
    mean_query, cov_query = compute_statistics(queries)
    # Compute unnormalized attention scores (z_i)
    scores = matmul(mean_query, keys.T) / math.sqrt(d)
    scores += einsum("i,ij,j->", keys, cov_query, keys) / (2 * d)
    # Normalize scores and weight by value norms
    scores = softmax(scores, dim=-1) * values.norm(dim=-1)
    # Keep KV pairs with highest scores
    n_kept = int(keys.size(0) * (1 - compression_ratio))
    indices = scores.topk(n_kept, dim=-1).indices
    return keys[indices], values[indices]
```

Listing 4.1. Pytorch-like pseudo code for KV Cache compression with Expected Attention.

Expected Attention Score With this query distribution, we can now analytically compute the expected unnormalized attention score in Equation 4.9. For a query $\bar{q} \sim \mathcal{N}(\bar{\mu}_q, \bar{\Sigma}_q)$ in our interval T and a fixed key k_i, the expected unnormalized score for that key is:

$$\hat{z}_i = \mathbb{E}_{\bar{q} \sim \mathcal{N}(\bar{\mu}_q, \bar{\Sigma}_q)} \left[\exp\left(\frac{\bar{q}^T k_i}{\sqrt{d}} \right) \right] = \exp\left(\frac{\bar{\mu}_q^T k_i}{\sqrt{d}} + \frac{k_i^T \bar{\Sigma}_q k_i}{2d} \right) \tag{4.14}$$

where the second equality follows from the moment-generating function of a Gaussian distribution. We then define the expected attention score by applying the softmax on our unnormalized expectation:

$$\hat{a}_i = \frac{\hat{z}_i}{\sum_{j=1}^{t} \hat{z}_j} \tag{4.15}$$

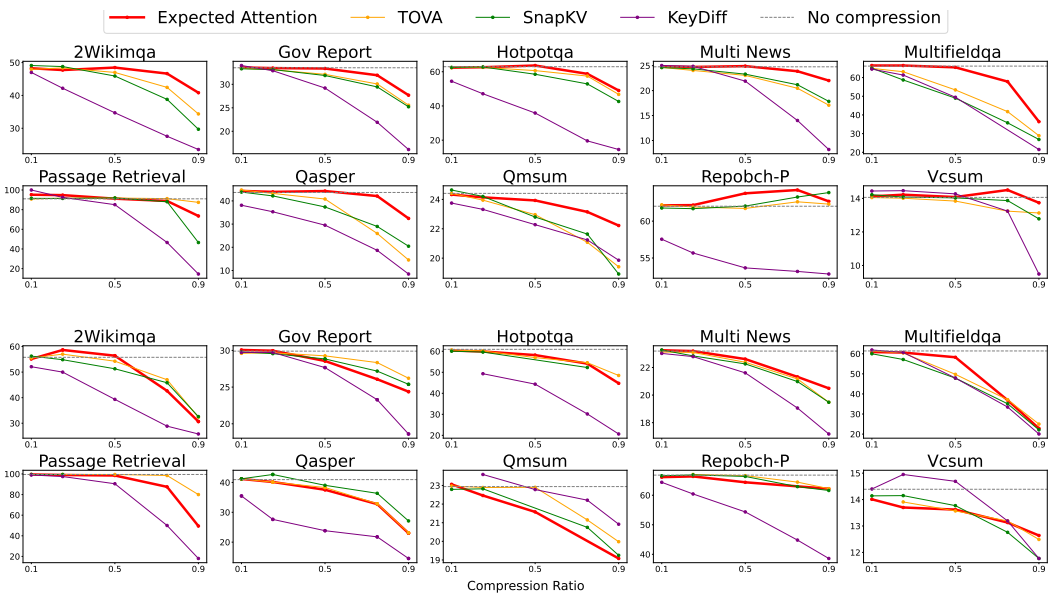


Figure 4.42. Scores on LongBench [36] for Qwen3-8B (top) and Gemma3-12B (bottom). The x-axis represents the compression ratio, the y-axis the score for each specific dataset. The horizontal line represents the baseline performance without cache compression. Expected Attention achieves optimal trade-off between compression ratio and scores across most datasets (Additional and averaged results in Section 4.3.9).

With this approximation, we can now estimate the importance of each cached KV pair. We define the expected contribution magnitude by substituting our expected attention weight into the contribution score formula from Equation 4.11:

$$\|\widehat{\Delta h_i}\| = (\hat{a}_i + \epsilon)\|W_o v_i\| \tag{4.16}$$

where $\hat{a}_i$ is the expected attention weight from Equation 4.15, $\|W_o v_i\| \in \mathbb{R}$ is the magnitude of the transformed value vector, and ϵ is a small hyperparameter. This metric provides a tractable approximation to the true contribution score without requiring future queries.

Compression with Expected Attention Equation 4.16 captures the contribution of each KV pair to the transformer output. The Expected Attention compression algorithm scores all cached KV pairs according to Equation 4.16 and evicts the r% pairs with the lowest expected contributions, where $r \in [0, 1]$ is the compression ratio. Intuitively, this is equivalent to removing those KV pairs that have the smallest impact on the residual stream and therefore on the model output. We provide pseudo-code for our compression algorithm in Listing 4.1.

Head-Adaptive Compression Previous work has shown that different attention heads serve different roles in the model. We adopt adaptive per-layer compression [184] to account for this heterogeneity, allowing more important heads to retain more KV pairs.

4.3.2 Experiments

Experimental Setup

Prefilling vs Decoding Generation LLM inference comprises two phases with distinct computational characteristics. The *prefilling phase* processes the entire input prompt in parallel, computing key-value projections for the KV Cache, a compute-bound operation requiring substantial floating-point operations. The *decoding phase* sequentially generates tokens using the KV Cache and previous logits, appending new key-value pairs iteratively [136, 215]. This dichotomy has motivated disaggregated architectures that implement prefill and decoding on different hardware [137, 548], at the cost of transferring the cache, further incentivising compression. An effective compression method must perform well in both prefilling and decoding [136, 215]. Nevertheless, a number of recent methods often target a single phase: SnapKV [349] for prefilling via query attention scores, StreamingLLM [622] and KNorm [158] for streaming decoding. Expected Attention is designed considering these two aspects of LLM inference and addresses both scenarios efficiently. We present results for prefilling and decoding in Section 4.3.3 and Section 4.3.3 respectively.

Models and Datasets For prefilling (one-shot compression before generation), we test three model families supporting long contexts: Llama3.1-8B (128k) [399], Qwen3-8B (32k) [640], and Gemma3-12B (128k) [199], all instruction-tuned. For decoding (compression during generation), we analyse reasoning models that generate extensive intermediate reasoning tokens and therefore large KV caches: Qwen-1.5B-R1, Qwen-7B-R1 [140], and OpenMath-Nemotron-14B [412].

Our benchmarks include LongBench [36], Ruler [264], and Needle in a Haystack [291, 366] for prefilling, and Aime25 [39] and MATH-500 [354] for decoding.

Baselines Following an initial benchmarking study on Ruler (see Section 4.3.9), we selected and compare our method against the best-performing baselines for each use case. For prefilling, we evaluate attention-based approaches like SnapKV [349] and TOVA [440], embedding-based KeyDiff [452], and the trainable DuoAttention [621] when the checkpoint is available. SnapKV [349] and TOVA [440] rank KV pairs using attention scores from user queries. KeyDiff [452] employs distance metrics between key embeddings for selection, making it also suitable for decoding generation. DuoAttention [621] takes a trainable approach, learning compression masks for each attention head. For decoding, we focus on methods designed to be compatible with streaming generation: KNorm [158], StreamingLLM [622], and KeyDiff [452]. KNorm [158] uses a simple approach by preserving keys with the lowest L_2 norm. StreamingLLM [622] maintains initial sink tokens throughout generation.

Implementation details We implement Expected Attention in Pytorch [456]. For all benchmarks, we test the models on 8 H100 GPUs, with batch size 1. We make all the code to reproduce our method and the baselines available online. In all experiments we use $\epsilon = 0.02$, except for needle in a haystack where use $\epsilon = 0$, and we average the RoPE embeddings over the next $T = 512$ positions. For prefilling, we do not assume any question about the context. This simulates a real world use case and avoids favouring

Table 4.3. Expected Attention outperforms most baselines on Ruler [264] with 4K and 16K context length. We show average score with increasing compression ratios across baselines. Best results for each compression ratio are displayed in **bold**. The 0% column indicates the baseline without compression.

Model	Method	Ruler 4k						Ruler 16k					
		0%	10%	25%	50%	75%	90%	0%	10%	25%	50%	75%	90%
Qwen3-8B	EA (ours)	**95.3**	**95.3**	**95.0**	**94.7**	**88.3**	**65.4**	**92.9**	**93.1**	**93.2**	**92.7**	**85.6**	**62.7**
	TOVA[440]	**95.3**	89.0	82.5	77.6	62.4	24.7	**92.9**	88.3	81.7	76.2	68.7	52.4
	SnapKV[349]	**95.3**	92.6	84.0	55.7	33.1	19.2	**92.9**	90.1	81.5	62.8	41.7	26.8
	KeyDiff[452]	**95.3**	93.8	89.4	78.6	64.4	37.9	**92.9**	88.9	82.9	74.5	66.9	53.1
Gemma3-12B	EA (ours)	**95.2**	**95.2**	**94.9**	**92.7**	**78.2**	**53.6**	**86.0**	**82.8**	**81.7**	**76.6**	**60.5**	**41.8**
	TOVA[440]	**95.2**	89.7	81.1	76.5	58.1	25.3	**86.0**	79.7	72.6	62.5	46.8	32.7
	SnapKV[349]	**95.2**	82.9	72.0	54.8	40.3	30.1	**86.0**	74.1	62.8	46.4	37.3	31.4
	KeyDiff[452]	**95.2**	94.3	90.6	79.8	62.0	34.3	**86.0**	81.8	78.6	72.6	58.6	37.2
Llama3.1-8B	EA (ours)	**95.3**	**95.7**	95.3	92.2	**75.9**	30.6	**93.4**	**93.4**	92.8	86.0	66.4	25.5
	TOVA[440]	**95.3**	93.2	87.3	76.2	63.3	37.5	**93.4**	90.9	86.1	77.9	68.4	59.2
	Duo [621]	**95.3**	95.7	**95.7**	**95.3**	73.2	24.5	**93.4**	93.3	**93.0**	**90.1**	59.1	12.3
	SnapKV[349]	**95.3**	95.5	88.8	81.8	63.2	43.4	**93.4**	89.4	82.0	68.0	43.1	25.6
	KeyDiff[452]	**95.3**	94.7	91.6	85.5	72.9	**61.1**	**93.4**	92.1	88.4	82.6	**74.9**	**66.5**

methods like SnapKV that rely on this assumption. For decoding, we keep a small buffer of hidden states of 128 tokens to compute statistics, and perform compression every 512 generation steps. In Equation 4.16 we only use V instead of $W_o V$, as using W_o led to a minor increase in results at a significantly higher memory cost.

4.3.3 Experimental Results

Prefilling

LongBench We evaluate on LongBench [36], which tests long-context capabilities across diverse tasks. The benchmark comprises six categories: single and multi-document QA, summarization, few-shot learning, synthetic tasks, and code completion. As shown in Figure 4.42 for Llama3.1-8B and Qwen3-8B (see Section 4.3.9 for Gemma3-12B), Expected Attention consistently achieves optimal compression-performance trade-offs, maintaining higher scores across all compression ratios. This demonstrates effective retention of critical KV pairs even under significant compression across varied reasoning and generation tasks.

Ruler Ruler [264] measures retrieval, multi-hop tracing, and aggregation abilities within long contexts through four subsets: NIAH (Needle-in-a-Haystack) for single-fact retrieval, VT (Variable Tracking) for multi-hop reasoning, CWE (Common Words Extraction) for frequency-based aggregation, and FWE (Frequent Words Extraction) for statistical pattern recognition. Table 4.3 shows results at various compression ratios for 4k and 16k windows. Expected Attention maintains strong performance across all subsets, particularly at higher compression ratios. While KeyDiff performs well on Llama3.1-8B, it struggles on Gemma3-12B and Qwen3-8B, potentially due to QK normalization [199, 640]. Our Expected Attention-based policy effectively preserves information necessary for precise retrieval and complex reasoning tasks.

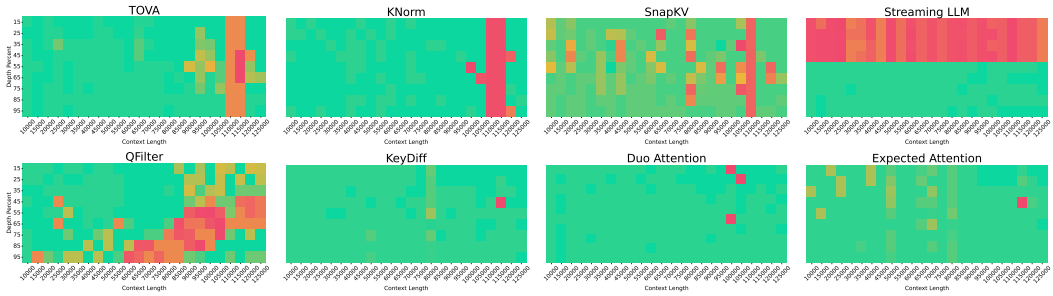


Figure 4.43. Needle in the Haystack test for different methods with Llama3.1-8B and 50% compression ratio.

Figure 4.44. Decoding results on Aime25 dataset, different markers represent different models sizes. The x-axis is the maximum size that the KV Cache is allowed to grow to.

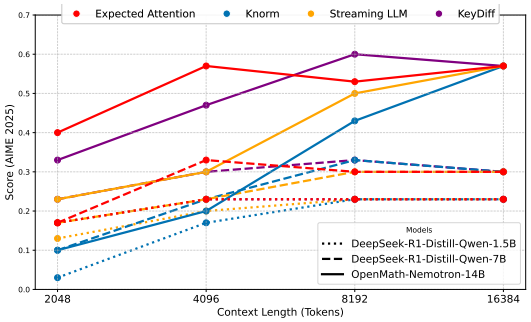


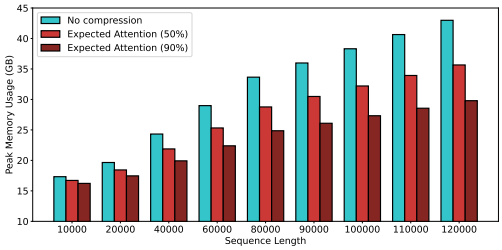
Table 4.4. Decoding scores on MATH-500. Columns indicate the final size of the KV Cache with respect to the original full version. Best scores in **bold**.

Model	Method	Compression			
		0×	2×	4×	12×
Qwen-R1-1.5B	EA (ours)	**0.47**	**0.47**	**0.43**	**0.33**
	KeyDiff[452]	**0.47**	0.42	0.40	0.30
	KNorm[158]	**0.47**	0.41	0.28	0.11
	Streaming[622]	**0.47**	0.45	0.41	0.31
Qwen-R1-7B	EA (ours)	**0.57**	**0.55**	**0.53**	**0.49**
	KeyDiff[452]	**0.57**	0.54	0.48	0.35
	KNorm[158]	**0.57**	0.47	0.32	0.12
	Streaming[622]	**0.57**	0.54	0.51	0.41
Nemotron-14B	EA (ours)	**0.57**	0.55	**0.54**	**0.47**
	KeyDiff[452]	**0.57**	0.56	0.51	0.44
	KNorm[158]	**0.57**	0.50	0.36	0.14
	Streaming[622]	**0.57**	**0.57**	**0.54**	0.42

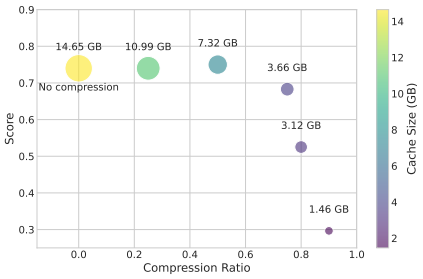
Needle in a Haystack The NIAH test [291] embeds specific information (the "needle") within lengthy distracting text (the "haystack") to evaluate retrieval capabilities across varying context positions and lengths. The test systematically varies both the needle's position within the context (needle depth) and the total context length to assess consistent retrieval performance. Figure 4.43 visualizes retrieval success across needle positions and context lengths up to 125k tokens. Expected Attention demonstrates robust performance comparable to DuoAttention and significantly more stable than other baselines in long contexts, confirming retention of critical information under compression regardless of needle placement or context size.

Decoding

We evaluate Expected Attention on reasoning models, Qwen-1.5B-R1, Qwen-7B-R1, and OpenMath-Nemotron-14B. Reasoning models are particularly suitable for our evaluation as they generate extensive chain-of-thought outputs, placing significant demands on KV Cache memory [703]. We use the Aime25 [639] and MATH-500 [354] datasets. Aime25 consists of competition-level mathematical problems requiring multi-step reasoning and precise calculation, while MATH-500 encompasses diverse mathematical domains including algebra, geometry, and number theory with varying difficulty levels. During



(a) Peak memory usage vs sequence length up to 120k for Llama3.1-8B, with 50% and 90% compression ratio. As the context length grows the memory savings become more evident, achieving up to 15GB less memory for large contexts.



(b) Needle in a Haystack score with different compression ratios with Qwen3-8B. Expected Attention has no accuracy loss with a compression ratio of 50%. Marker size indicates actual KV Cache size in GB.

decoding, we allow the KV Cache to expand to a predetermined size before initiating token eviction. We use $n\times$ to show that the final cache size is n times smaller than would be without compression.

Results for Aime25 and MATH-500 are presented in Figure 4.44 and Table 4.4, respectively. Expected Attention consistently outperforms or matches baseline methods across all models, with particularly strong performance at higher compression ratios (4× and 16×). Most methods demonstrate minimal performance degradation at 2× compression, indicating that a large portion of tokens in reasoning traces contains redundant information that can be pruned without affecting mathematical reasoning performance. Expected Attention shows the best performance especially in high-compression scenarios (12× compression).

Memory Savings and Efficiency

We evaluate the memory efficiency of our method using Llama3.1-8B and Qwen3-8B for both prefilling and decoding phases. All experiments are conducted on a single H100 GPU with bfloat16 precision for both model weights and KV Cache. We focus on peak memory usage as the primary efficiency metric, as KV Cache memory consumption is often the primary bottleneck for long-context inference.

Figure 4.45a demonstrates peak memory usage as sequence length increases up to 120k tokens, comparing Expected Attention at 50% and 90% compression ratios against the uncompressed baseline with vanilla attention. The results show that memory savings become increasingly substantial as context length grows.

Figure 4.45b illustrates the relationship between compression ratio (x-axis) and NIAH benchmark performance for Qwen3-8B, with marker size representing the corresponding KV Cache size. While higher compression ratios naturally reduce KV Cache size, they typically incur performance penalties. Remarkably, Expected Attention at 50% compression maintains performance parity with the uncompressed baseline while achieving a 2× reduction in KV Cache size, demonstrating an optimal balance between memory efficiency and task performance.

4.3.4 Related Work

Trainable KV-Cache Compression One approach to reducing memory requirements involves modifying the model architecture or training procedure to inherently produce smaller caches. [9, 530] reduce cache size by decreasing the number of key-value heads, effectively sharing key-value representations across queries. DeepSeek-V2 [139] introduced Multi-Head Latent Attention, which projects keys and values into a lower-dimensional latent space during training, directly reducing the memory footprint of cached representations. Alternative trainable approaches focus on learning compression policies [703, 423] or masks [621] from pre-trained checkpoints. Finally, State Space Models [226, 225] replace the quadratic attention mechanism with linear-complexity alternatives, while hybrid approaches combine transformer layers with RNN-based components [498, 207]. Although these trainable methods typically achieve superior performance, they require substantial computational resources for pre-training or continued pre-training, making them less practical for deployment with existing large-scale models.

Training-Free KV Cache compression Given the computational costs associated with trainable methods, significant research effort has focused on developing post-training compression techniques that can be applied to existing models without modification. Early approaches [349, 440] directly utilize attention scores to rank KV pairs by importance. However, these methods require access to the full attention matrix, making them incompatible with Flash Attention [131] and thus impractical for modern deployment scenarios. To address this limitation, several works have developed heuristic-based importance measures that can be computed without materializing attention matrices, such as keys norm (KNorm [158]), token positions (StreamingLLM [622], H2O [676]) or SVD projection (Q-Filters [210]). Recognizing that different attention heads exhibit varying sensitivity to compression, recent methods such as AdaKV [184] and PyramidKV [80] adopt head-specific compression strategies. *Expected Attention*, adopts insights from these heuristic approaches while providing a principled theoretical foundation based on the distributional properties of transformer activations.

Quantization Instead of reducing the KV Cache size along the sequence dimension, quantization methods try to reduce the precision used to store the cache. For example, NQKV [81] partitions the cache into blocks for quantization and processes them separately. KVQuant [258] performs non uniform per-layer quantization, while KIVI [696] quantizes the key cache by layer and the value cache by token. These methods are orthogonal to Expected Attention (and to KV Cache compression in general), making it possible to integrate them.

Efficient Implementations Alongside compression, sparse attention and quantization, another effort has been done to devise efficient implementation of inference systems. In this context, a well designed low-level handling of the KV Cache can deliver significant performance speed-ups, especially in multi-user serving systems. The first to investigate this and introduce efficient memory management for KV Cache was vLLM [318], soon followed by other approaches [473, 284] and frameworks [432].

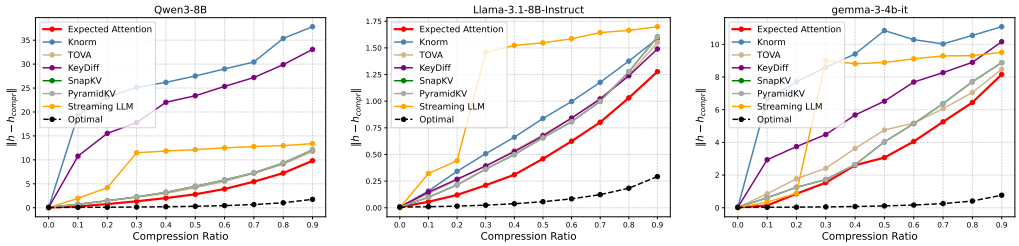


Figure 4.46. Reconstruction error $\|h - h_{\text{compr}}\|$ averaged across model layers. Expected Attention achieves the best error, minimizing the impact on the residual stream.

4.3.5 Limitations

A key trade-off of our training-free methodology is that its performance does not match that of trainable methods [138, 703]. This is an intentional design choice that allows deployment without significant computational resources required for intensive training. Future work could explore combining our theoretical framework with lightweight fine-tuning.

Another limitation is that our method requires users to specify compression ratios manually, lacking an automated mechanism to determine optimal compression levels for different scenarios such as text generation. This represents a promising area for future research.

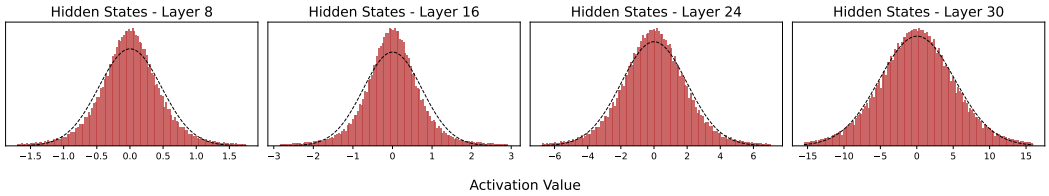
Finally, while our PyTorch implementation effectively demonstrates our method's theoretical principles, it is not optimized for efficiency. A highly performant implementation with custom CUDA kernels would significantly improve speed and practical utility.

4.3.6 Reconstruction Error Across Methods

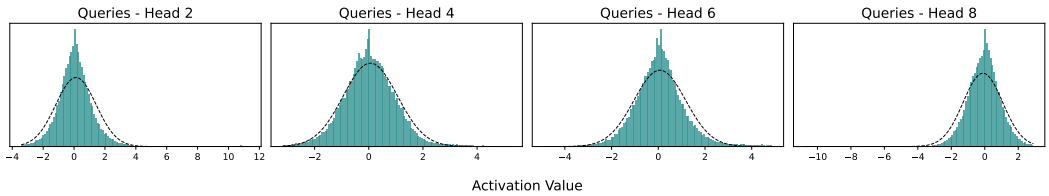
In Section 4.3.1, we discussed the challenge of compressing the KV Cache without significantly altering the residual stream. To understand the impact of Expected Attention on the model output, we quantify the reconstruction error of the residual stream, i.e. how the difference between the original, uncompressed hidden states and the corresponding hidden states after compression. We define the reconstruction error as $\|h - h_{\text{compr}}\|$, where h is the original hidden state without compression and h_{compr} the hidden state after the KV Cache has been compressed. We average the reconstrcution error over a long sequence of $\sim$ 5K tokens and display the results for several methods in Figure 4.46. Expected Attention consistently achieves a lower reconstruction error, indicating that it preserves the integrity of the hidden state more effectively than competing methods, a crucial property for maintaining downstream performance [413, 215].

4.3.7 Distributional Properties of LLM activations

In this section, we analyse the distributional properties of activations within Large Language Models. Our investigation aligns with the findings of prior work, which has demonstrated that LLM activations often exhibit normal distributions. More specifically [364] finds that hidden states are zero-mean unimodal, and qualitatively fall into two



(a) Qwen3-8B Hidden States distributions.



(b) Qwen3-8B queries distributions.

Figure 4.47. Distributions of Qwen3-8B Hidden States and queries.

distinctly shaped distributions. The hidden states before the Attention and the MLP layers tend to be Gaussian-like, while the hidden states in the intermediate of such layers tend to be Laplacian-like.

For Expected Attention, we are interested in the hidden states before the MLP layers and the corresponding queries. Our study confirms that such activations are predominantly unimodal and can be approximated as Gaussian distributions, albeit with the presence of a few heavy-tailed outliers, as already found in [622, 556]. In Figure 4.49a, Figure 4.48a, and Figure 4.47a we show hidden states and queries for different models. For our method, the distributional properties of queries are of particular importance, and we observe that queries maintain a clear Gaussian-like behaviour. This also applies to models with QK normalization, where the query projection is not guaranteed to be linear. The concentration of these activations around a central value and their Gaussian like shape provides the theoretical basis for Expected Attention.

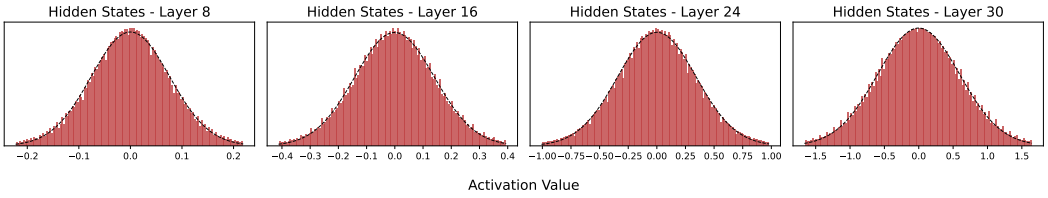
We stress that in this work, our goal is not to explain or investigate this property, but rather to leverage it for KV Cache compression.

4.3.8 Expected Attention Score

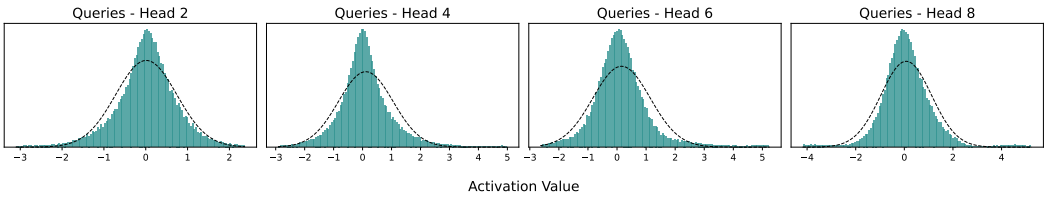
To empirically validate that the expected attention score is strongly correlated to the real model attention score, we plot the correlation between the observed attention and the expected attention score across different layers and heads. We use sequence of 5K tokens and use the first 1K tokens to compute the query statistics. We display the results in Figure 4.50. We see that for different layers and attention heads, the expected attention score from Equation 4.11 is strongly correlated to the original attention score.

4.3.9 Additional Results

In Table 4.5 we show additional results on the LongBench dataset, averaged across all subsets.

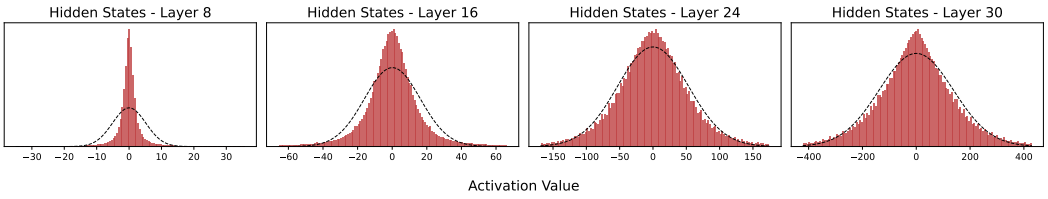


(a) Llama3.1-8B hidden states distributions.

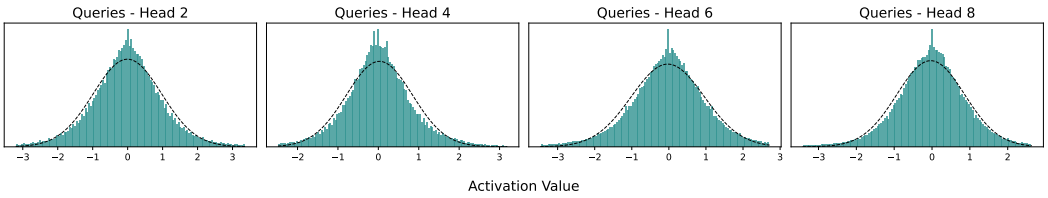


(b) Llama3.1-8B queries distributions.

Figure 4.48. Distributions of Llama3.1-8B hidden states and queries.



(a) Gemma3-12B hidden states distributions



(b) Gemma3-12B queries distributions.

Figure 4.49. Distributions of Gemma3-12B hidden states and queries.

Ruler In order to select the most competitive baselines we performed an initial search on 15+ methods on Ruler. We selected the best performing ones as displayed in Figure 4.51. We did not include KVZip [299] despite achieving a high score as it needs two forward passes, therefore implying a higher cost FLOPs that is double as much as the other baselines.

4.3.10 Conclusion

We introduced Expected Attention, a training-free algorithm for KV Cache compression. We showed Expected Attention outperforms state-of-art KV Cache compression methods on several benchmarks and in both prefilling and decoding scenarios. Additionally, we released a research library that allows researchers to easily implement and experiment with KV Cache compression methods, and evaluate them on popular benchmarks for long context.

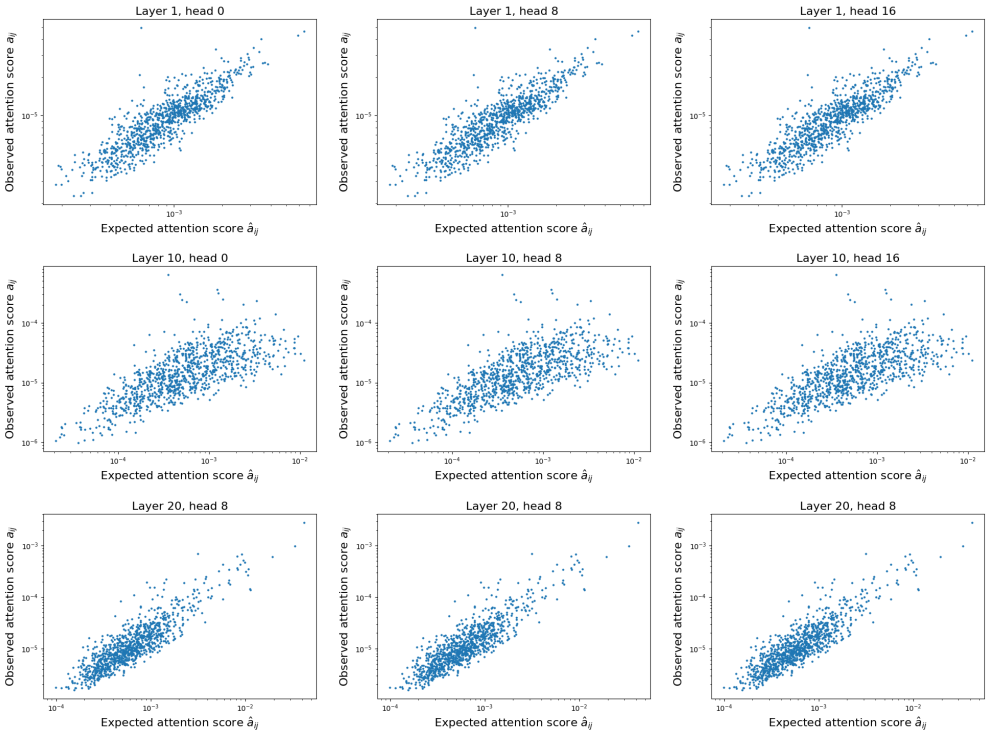


Figure 4.50. Correlation between attention score and expected attention score for Llama3.1-8B. We compute the expected attentions score on a sequence of 5K tokens, using the first 1K for statistics. A strong correlation exists between our attention score approximation and the observed attention score.

4.3.11 Reproducibility statement

To ensure the reproducibility of our work, we are providing a complete and self-contained codebase along with this submission. The provided code includes all necessary scripts for data preprocessing and evaluation, allowing for the direct replication of our experiments and results. For now, we share the repo in an anonymized github repository.

The codebase is organized to be straightforward to use and is accompanied by a `README.md` file with detailed instructions on how to set up the environment and run the experiments.

Table 4.5. Expected Attention outperforms most baselines on Longbench [36]. We show average score with increasing compression ratios across baselines.

Model	Method	Longbench					
		0%	10%	25%	50%	75%	90%
Qwen3-8B	Expected Attention	**48.63**	48.30	**50.25**	**50.1**	**48.06**	**39.71**
	TOVA	**48.63**	48.41	48.14	46.49	43.19	37.21
	SnapKV	**48.63**	**48.40**	47.85	46.25	42.42	34.57
	KeyDiff	**48.63**	48.13	46.23	40.08	29.42	20.69
Gemma3-12B	Expected Attention	**51.04**	**54.02**	50.98	47.51	40.41	32.67
	TOVA	**51.04**	53.05	**51.52**	**50.7**	**46.88**	**40.45**
	SnapKV	**51.04**	51.83	51.31	48.14	44.31	34.97
	KeyDiff	**51.04**	51.64	48.74	42.15	33.68	23.46
Llama3.1-8B	Expected Attention	**46.42**	**46.59**	**46.8**	**47.91**	**44.04**	33.97
	TOVA	**46.42**	46.22	45.62	44.13	40.5	34.77
	SnapKV	**46.42**	46.56	46.07	45.07	41.24	32.55
	KeyDiff	**46.42**	46.45	48.01	46.9	42.24	**35.51**

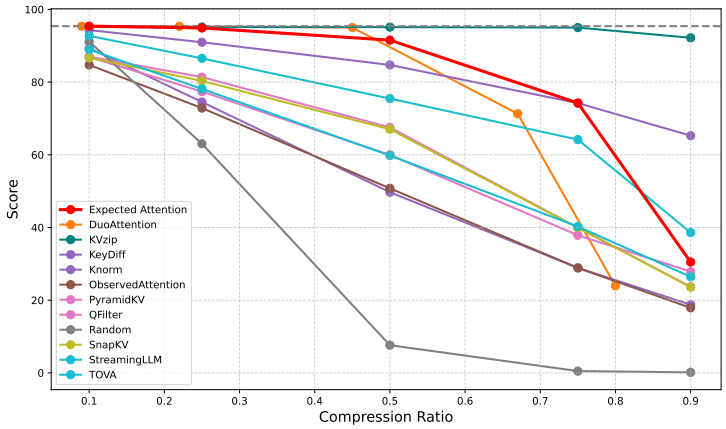


Figure 4.51. Initial experiments on Ruler 4K to select the best baselines. We did not use KVZip as it requires two forward passes and increases latency significantly.

Chapter 5

Interpretable AI Systems

5.1 Spare: SAE-based representation engineering for solving knowledge conflicts

Large language models (LLMs) have shown remarkable capability to memorise factual knowledge and solve knowledge-intensive tasks [460, 75, 575, 282, 16]. Nevertheless, the knowledge stored in their parameters (*parametric knowledge*) can be inaccurate or outdated [633]. To alleviate this issue, retrieval and tool-augmented approaches have been widely adopted to provide LLMs with external knowledge (*contextual knowledge*) [295, 333, 616, 523]. However, contextual knowledge may sometimes conflict with the parametric knowledge of the model, leading to what we refer to as *knowledge conflicts*. Such conflicts can cause undesired behaviour, where the model may rely on inaccurate information sources, resulting in incorrect outputs [379, 626, 554, 599, 677].

Prior research found that LLMs tend to prefer contextual knowledge (*e.g.,*, retrieved passages) over their parametric knowledge when conflicts occur [554, 626, 255]. For instance, [554] show that most LLMs choose parametric knowledge in less than 10% examples. However, in more general applications, LLMs should retain the ability to use their parametric knowledge when presented with misinformation [94, 93, 701, 379, 684]. Existing works investigate fine-tuning and prompting-based strategies to detect and resolve knowledge conflicts [599]; however, they need additional interactions with the model, *e.g.,*, by asking the LLMs to examine the conflicts sentence by sentence, resulting in high latency times and preventing practical applications.

In this work, we investigate *representation engineering* methods to efficiently steer the usage of parametric and contextual knowledge of LLMs at *inference time*. Although representation engineering has provided an efficient and transparent framework for controlling the behaviour of LLMs, we find that existing methods fail to effectively steer knowledge usage. This may be because these methods directly modify the internal activations of LLMs, such as hidden states [580, 699] or MLP activations [481, 393]. These activations are polysemantic dense vectors that overlap with many independent semantic features [433]. Thus, minor edits in one dimension can influence multiple semantic features, making it difficult to adjust activations accurately without affecting other features in practice.

Recently, sparse auto-encoders (SAEs) have been proposed to address the difficulty of interpreting polysemantic activations by decomposing them into a large-scale monose-

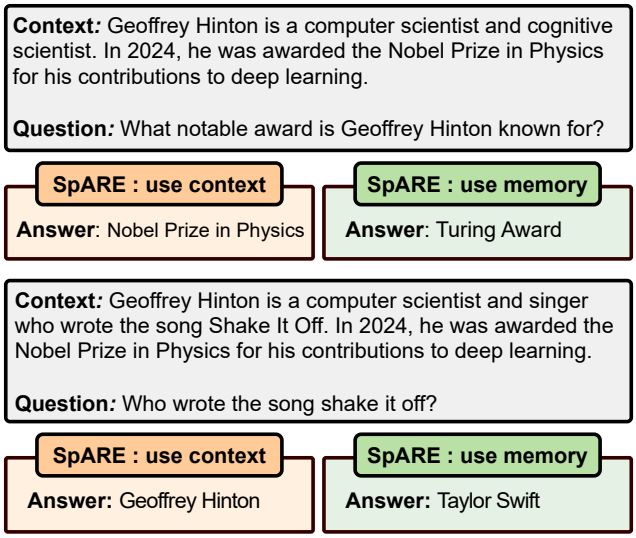


Figure 5.1. In the event of a knowledge conflict, the model can rely on the context or on the parametric knowledge. The figure presents the predictions of Llama2-7B steered by SPARE.

mantic feature dictionary [271, 194, 566]. Therefore, we introduce SAEs as a tool for precise activation editing to guide the knowledge selection of LLMs. Specifically, we propose SPARE, a **Sp**arse **A**uto-Encoder-based **R**epresentation **E**ngineering method to steer the knowledge selection behavior of LLMs. SPARE first identifies the SAE activations that are related to specific knowledge selection behaviours (Section 5.1.3); then, it extracts functional features that control the usage of contextual and parametric knowledge, and finally applies them to steer the behaviour of the model (Section 5.1.3).

Our experimental results on open-domain question-answering tasks show that SPARE effectively controls the knowledge selection behaviours by utilising a small set of SAE features, e.g., less than 0.05% SAE activations for Gemma2-9B in the 6 layers[1]. SPARE yields more accurate results than state-of-the-art representation engineering methods (+10%), contrastive decoding (+15%), and in-context learning (+7%), achieving the best performance on steering knowledge selection behaviours of LLMs under knowledge conflicts.

5.1.1 Background

Problem Setup Following [372, 255, 626], we use open-domain question-answering (ODQA) tasks to investigate the behaviour of LLMs when there is a conflict between the parametric knowledge of the model and contextual knowledge. In ODQA datasets with knowledge conflicts, each instance is presented as (Q, E_M, M, E_C, C), where Q is the question, E_M is the evidence that supports the memorised knowledge stored in the model parameters, E_C is the evidence that conflicts with the language model's memorised knowledge, M is the answer based on the E_M, and C is the answer based on the E_C.

[1]For Gemma2-9B, we use the pre-trained SAEs from GemmaScope https://huggingface.co/google/gemma-scope, and the selected activations is presented in Section 5.1.10.

Sparse Auto-Encoders Recent works have proposed using sparse auto-encoders (SAEs) to interpret the complex representations of LLMs by decomposing them into a large set of monosemantic features [271, 194, 566]. Given an activation $\mathbf{h} \in \mathbb{R}^d$ from the residual stream of LLMs, a SAE with n latent dimensions encodes it into a sparse vector $\mathbf{z} \in \mathbb{R}^n$ and decodes it to recover $\mathbf{h}$:

$$\begin{aligned} f_\theta(\mathbf{h}) &= \sigma\left(\mathbf{W}_\theta\left(\mathbf{h}-\mathbf{b}\right)+\mathbf{b}_\theta\right)=\mathbf{z}, \\ g_\phi(\mathbf{z}) &= \mathbf{W}_\phi\ \mathbf{z}=\sum_{i=1}^{n} z_i\mathbf{f}_i+\mathbf{b}=\hat{\mathbf{h}} \end{aligned} \tag{5.1}$$

where σ is an activation function that outputs a non-negative value such as ReLU, $\mathbf{W}_\theta \in \mathbb{R}^{n\times d}$, $\mathbf{b} \in \mathbb{R}^d$, $\mathbf{b}_\theta \in \mathbb{R}^n$, $\mathbf{W}_\phi \in \mathbb{R}^{d\times n}$, z_i is the i-th element of the SAE activation $\mathbf{z}$, and $\mathbf{f}_i \in \mathbb{R}^d$ is the i-th column of $\mathbf{W}_\phi$. The $\{\mathbf{f}_i\}_{i=1}^n$ learned through the SAE are considered highly monosemantic, and the SAE activation $\mathbf{z}$ indicates the activated values of $\{\mathbf{f}_i\}_{i=1}^n$.

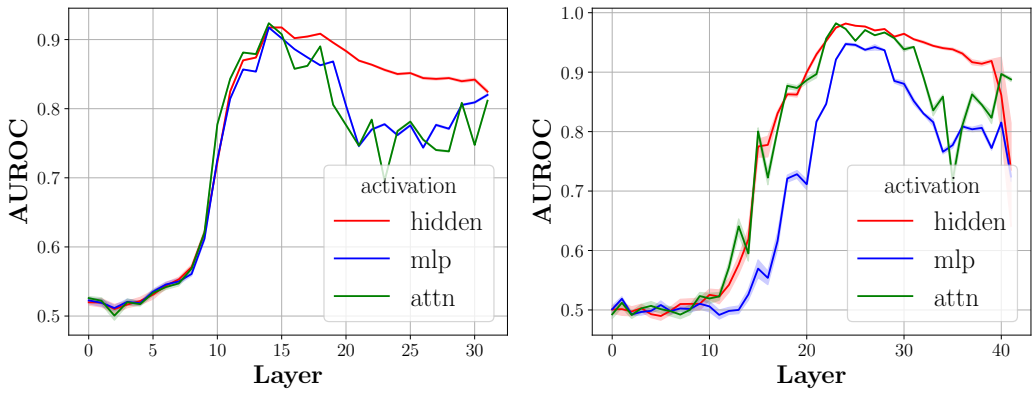
5.1.2 Detection of Knowledge Conflicts

In this section, we investigate whether we can detect the occurrence of conflicts during the generation process, since identifying such conflicts is a prerequisite for exploring inference-time strategies to control the LLM.

We focus on the residual stream [176] of the model and look for a signal of knowledge conflict. To this end, we create two groups of input instances, $\mathcal{D}_{E_M} = \{(Q, E_M)\}$ and $\mathcal{D}_{E_C} = \{(Q, E_C)\}$. In $\mathcal{D}_{E_M}$, the model is provided with a context that is coherent with the model internal memorized knowledge, whereas in $\mathcal{D}_{E_C}$ the model is provided with a context that does not agree with model parametric knowledge, thus causing a knowledge conflict. To determine whether a signal of conflict arises in the residual stream, we focus on the last position of the sequence during generation, which is supposed to encode the information to predict the first token of the answer.

We apply a linear probing method [116, 695, 11] to investigate whether the residual stream contains a signal of knowledge conflict. Specifically, we train logistic regression models to classify whether a given activation (the hidden state, MLP or Self-Attention activations) is from the $\mathcal{D}_{E_C}$ or $\mathcal{D}_{E_M}$, i.e. whether it contains a knowledge conflict or not. We use activations from each layer as input and formulate this as a binary classification task. The evaluation is conducted on a held-out test set. We present probing results on Llama2-7B [575] and Gemma2-9B [502] using AUROC as metric in Fig. 5.2. We observe that the probing accuracy increases from the first layer to the middle layers, and this trend is the same across different types of activations. This indicates that we can detect the signal of knowledge conflict in the residual stream of the mid-layers. The probing accuracy decreases in the later layers, especially for MLP and Self-Attention activations, which indicates that MLP and Self-Attention modules do not further add the signal of conflicting knowledge to the residual stream. We provide more details and analysis about knowledge conflict detection in Section 5.1.7 and [679].

The above analysis shows that knowledge conflicts can be identified in the internal states of LLMs. Moreover, it provides insight into which layers can be more influential in the knowledge selection (Section 5.1.5).



(a) Llama2-7B **(b)** Gemma2-9B

Figure 5.2. The knowledge conflict probing results of Llama2-7B and Gemma2-9B on NQSwap [372]. The probing results on hidden states, MLP and Self-Attention activations are coloured differently.

5.1.3 Resolving Knowledge Conflicts by Representation Engineering

In this section, we introduce SPARE, our SAE-based representation engineering method, to steer the usage of parametric and contextual knowledge to generate the answers. SPARE consists of the three following steps: *1)* collecting activations that lead to different knowledge selection behaviours (Section 5.1.3); *2)* identifying SAE activations that are related to each knowledge selection behaviour (Section 5.1.3); *3)* steering the usage of either knowledge source by editing the hidden states of LLMs at inference time (Section 5.1.3).

Collecting Activations with Different Knowledge Selection Behaviours

We showed in Section 5.1.2 that we can detect the knowledge conflict by probing the residual stream. We now want to characterise the activations that lead to different knowledge selection behaviours. To this end, given a set of instances $\mathcal{D}_{E_C}$ that cause a knowledge conflict, we separate it into two groups based on the model's predictions: $\mathcal{D}_C$, where the model generates an answer that aligns with the context, and $\mathcal{D}_M$, where the model ignores the context and generates an answer relying on the parametric knowledge. These two subsets characterise two knowledge selection behaviours of the model. In the following, we omit the notation to specify the layer of $\mathbf{h}$ and $\mathbf{z}$ for simplicity, as the method can be applied to arbitrary layers. We collect the hidden state at the last position of the input that is used to generate the first token of the answer.

We collect the hidden states from $\mathcal{D}_C$ and $\mathcal{D}_M$ for N samples, denoting them as $\{\mathbf{h}_C^j\}_{j=1}^N$ and $\{\mathbf{h}_M^j\}_{j=1}^N$, respectively. We then obtain the SAE activation for each sample by $\mathbf{z}_C^j = f_\theta(\mathbf{h}_C^j)$ and $\mathbf{z}_M^j = f_\theta(\mathbf{h}_M^j)$. Finally, we compute the average of the sets $\{\mathbf{z}_C^j\}_{j=1}^N$ and $\{\mathbf{z}_M^j\}_{j=1}^N$ to obtain the mean vectors $\overline{\mathbf{z}_C}$ and $\overline{\mathbf{z}_M}$, respectively. More details are presented in Section 5.1.9 and Section 5.1.9.

At this stage, $\overline{\mathbf{z}_C}$ and $\overline{\mathbf{z}_M}$ contain the information to steer the generation towards C or M. However, there might still be instance-specific activations with non-zero values

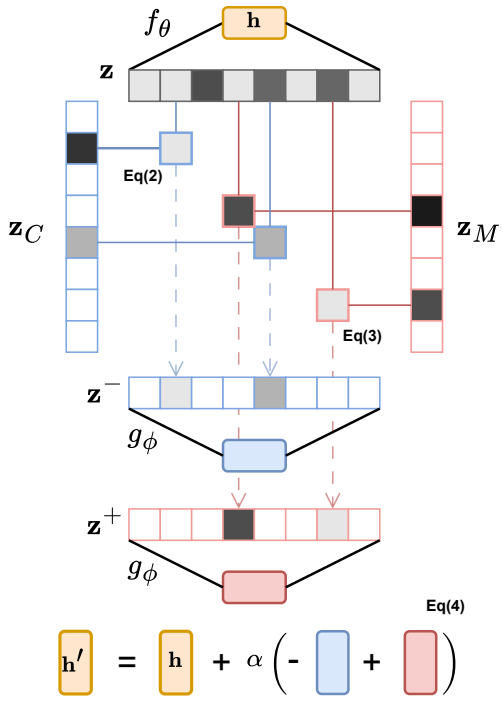


Figure 5.3. The workflow of SPARE steers the knowledge selection behaviour. The figure presents an example of steering the model to use parametric knowledge. First, the SAE encoder f_θ encodes hidden state $\mathbf{h}$ into the SAE activation $\mathbf{z}$. Then, it determines the values of SAE activations $\mathbf{z}^-$ and $\mathbf{z}^+$ for editing (Eq. (5.2) and Eq. (5.3)). Finally, we edit the hidden state using the features extracted from the SAE decoder g_ϕ (Eq. (5.4)).

that are not responsible for the knowledge selection behaviour. In the next section, we identify functional activations related to knowledge selection behaviours and then construct two orthogonal SAE activations, $\mathbf{z}_C$ and $\mathbf{z}_M$ for steering the knowledge selection behaviours.

Identifying Functional SAE Activations

As shown by previous works [194, 566], a single SAE activation can capture one monosemantic feature. In this work, we hypothesise a combination of a small set of SAE activations can be responsible for a functional feature, such as knowledge selection in case of conflict. Our hypothesis is motivated by Task Vector [248, 570], which shows that hidden states contain the functional information that drives a task.

We now show how we find the SAE activations that are responsible for driving the knowledge selection. First, we calculate mutual information between each SAE activation and the knowledge selection behaviours, which measures to which extent the behaviour depends on each activation. Let the random variable Z_i be the ith activation of SAE, and $Y = \{C, M\}$ be the generated answers; we calculate the mutual information $I(Z_i; Y)$ between them. A higher $I(Z_i; Y)$ indicates a higher dependency between Z_i and the knowledge selection behaviour. We then select the top-k activations with the highest $I(Z_i; Y)$, denoted as $\mathcal{Z}$. More details are available in Section 5.1.9

In the following, we determine which knowledge selection behaviour each $Z_i \in \mathcal{Z}$ positively correlates with. Given the sets of activations $\{\mathbf{z}_C^j\}_{j=1}^N$ and $\{\mathbf{z}_M^j\}_{j=1}^N$, we estimate the expected value of each activation feature $Z_i \in \mathcal{Z}$ in both sets, denoted as $\mathbb{E}_C[Z_i]$ and $\mathbb{E}_M[Z_i]$. We then have that Z_i is positively correlated with the behaviour of selecting contextual knowledge if $\mathbb{E}_C[Z_i] - \mathbb{E}_M[Z_i] > 0$. Conversely, if this condition is not met, Z_i is positively correlated with the behaviour of selecting parametric knowledge. Finally, we construct two functional SAE activations $\mathbf{z}_C$ and $\mathbf{z}_M \in \mathbb{R}^n$, that steer the usage of contextual and parametric knowledge, respectively. For each element, z_{Ci} and z_{Mi} are set to 0 if $Z_i \notin \mathcal{Z}$, and the remaining values are taken from $\overline{\mathbf{z}_C}$ and $\overline{\mathbf{z}_M}$ based on their expectations:

$$z_{Ci} = \begin{cases} \overline{z_{Ci}}, & \text{if } \mathbb{E}_C[Z_i] - \mathbb{E}_M[Z_i] > 0 \\ 0, & \text{otherwise} \end{cases}$$

$$z_{Mi} = \begin{cases} \overline{z_{Mi}}, & \text{if } \mathbb{E}_C[Z_i] - \mathbb{E}_M[Z_i] < 0 \\ 0, & \text{otherwise} \end{cases}$$

Editing Activations to Steer Behaviours

In the following, we introduce how we utilise the functional activation $\mathbf{z}_C$ and $\mathbf{z}_M$ to control the usage of knowledge sources at inference time. Suppose we want to control the LLM to use its parametric knowledge and ignore the conflict contextual knowledge that might be misinformation. In this case, we aim to *remove* the features that steer the contextual knowledge usage and *add* the features that steer the parametric knowledge usage. To avoid removing or adding unnecessary features, we restrict the values to edit by the following two constraints. Let an activation be $\mathbf{h}$ and corresponding SAE activations be $\mathbf{z} = f_\theta(\mathbf{h})$. First, in Eq. (5.2), we determine the value we need to remove from z_i to avoid the undesired behaviour, i.e., generating contextual knowledge in this case. At this step, we ensure that the resulting activation remains non-negative after the removal, i.e., subtract at most z_i when $z_i < z_{Ci}$:

$$z_i^- = \min\{z_i, z_{Ci}\}. \tag{5.2}$$

Then, in Eq. (5.3), we determine the value we need to add to z_i to encourage the desired behaviour, i.e., generating parametric knowledge in this case. Here, we ensure that no excess value is added once the activation reaches z_{Mi}:

$$z_i^+ = \max\{z_{Mi} - z_i, 0\}. \tag{5.3}$$

Finally, we obtain the edited hidden states $\mathbf{h}'$ by:

$$\mathbf{h}' = \mathbf{h} + \alpha\left(-g_\phi\left(\mathbf{z}^-\right) + g_\phi\left(\mathbf{z}^+\right)\right), \tag{5.4}$$

where $\alpha \in \mathbb{R}^+$ is a user-defined hyperparameter that controls the degree of editing. Note that we do not directly edit the activation $\mathbf{z}$ of hidden state $\mathbf{h}$ to obtain the modified hidden states by $\mathbf{h}' = g_\phi(\mathbf{z}')$ with $\mathbf{z}' = \mathbf{z} - \mathbf{z}^- + \mathbf{z}^+$ for two reasons: *1)* it can result in unexpected information loss of original $\mathbf{h}$ due to the reconstruction loss of the SAE, leading to a nearly zero accuracy in our experiments; *2)* the definition of the SAE

activation shown in Eq. (5.1) requires each element of $\mathbf{z}$ be a non-negative value, which prevents us from using α to flexibly control the strengthen of editing like Eq. (5.4).

Similarly, if we want to control the model to be faithful to the context in the case of the contextual knowledge is more likely correct, we can swap z_{Ci} and z_{Mi} in Eq. (5.2) and Eq. (5.3). In this work, we only edit hidden states at the last position of the input for ODQA tasks.

5.1.4 Experimental Results

Settings

Datasets We use two widely adopted open-domain question-answering datasets with knowledge conflicts NQSwap [372] and Macnoise [255] to investigate the controlling capability of several methods.

Models We evaluate our method using Llama3-8B [399] and Gemma2-9B [502], which have corresponding public pre-trained SAEs. Moreover, we also evaluate our method using Llama2-7B [575] with our pre-trained SAEs to examine the feasibility of adopting SPARE to an LLM without public SAEs. More details are presented in Section 5.1.8.

Evaluation We use the greedy decoding method to evaluate the LLMs in an open-ended generation setting. We use 3 in-context demonstrations to align the answer format. The demonstrations use non-conflict evidence E_M and memorised answer M, so they do not point out which knowledge source to select. More details are presented in Section 5.1.9. The test examples use E_C, leading to a knowledge conflict for LLMs, and a behaviour-controlling method needs to steer the usage of either parametric or contextual knowledge to generate the answer. We compare the evaluation results under control with the results without any control to show each method's controlling capability.

Baselines We compare SPARE against the following inference-time *representation engineering* methods: 1) TaskVec [248]; 2) ActAdd [580]; 3) SEA [481] with linear and non-linear versions, noted by subscript "linear" and "SqExp". We compare with the following *contrastive decoding* methods: 1) DoLa [111]; 2) CAD [535]. Moreover, we also compare using in-context learning (ICL) [75] to steer the knowledge selection. We use E_C and C in the demonstrations to guide the model to ignore its parametric knowledge and use the contextual knowledge, and use E_C and M to guide the model to ignore the contextual knowledge and use its parametric knowledge. ICL is not an inference-time strategy because it requires changing the original input of the model to achieve a desired behaviour. More details of baseline implementation and hyperparameters searching are presented in Section 5.1.9 and Section 5.1.9.

Hyperparameters We select the proper hyperparameters for SPARE in the developments set that is also used to select the hyperparameters of baselines, and the details are presented in Section 5.1.9. In the following, we apply SPARE from the 12th to the 15th and 13th to the 16th layer for Llama2-7B and Llama3-8B and from the 23rd to 25th and the 29th to 31st layers for Gemma2-9B; we analyse the performance of editing individual layers in Section 5.1.5.

Metric	Method	NQSwap [372]			Macnoise [255]		
		Llama3-8B	Llama2-7B	Gemma-2-9B	Llama3-8B	Llama2-7B	Gemma-2-9B
Steer to Use Parametric Knowledge							
EM_M	*Without Controlling*	$26.63_{\pm 6.02}$	$22.23_{\pm 4.75}$	$26.32_{\pm 1.80}$	$18.96_{\pm 2.65}$	$22.37_{\pm 1.89}$	$17.06_{\pm 3.79}$
	TaskVec [248]	$24.16_{\pm 6.58}$	$24.88_{\pm 0.85}$	$29.85_{\pm 0.83}$	$21.23_{\pm 1.89}$	$22.93_{\pm 2.31}$	$28.92_{\pm 1.19}$
	ActAdd [580]	$37.87_{\pm 8.96}$	$31.43_{\pm 3.68}$	$27.67_{\pm 0.82}$	$26.17_{\pm 0.22}$	$27.52_{\pm 3.07}$	$29.75_{\pm 1.68}$
	SEA_{linear} [481]	$21.03_{\pm 1.83}$	$23.73_{\pm 0.86}$	$24.43_{\pm 0.91}$	$12.84_{\pm 0.18}$	$15.64_{\pm 0.24}$	$28.10_{\pm 2.78}$
	SEA_{SqExp} [481]	$13.64_{\pm 1.62}$	$16.66_{\pm 0.55}$	$23.79_{\pm 1.38}$	$14.24_{\pm 1.45}$	$16.24_{\pm 1.06}$	$28.07_{\pm 1.30}$
	DoLa [111]	$25.53_{\pm 5.19}$	$16.50_{\pm 3.91}$	$20.58_{\pm 1.06}$	$16.52_{\pm 2.65}$	$15.66_{\pm 0.88}$	$19.81_{\pm 2.58}$
	$^{\flat}$CAD [535]	$33.72_{\pm 0.84}$	$31.23_{\pm 1.45}$	$41.17_{\pm 0.59}$	$28.58_{\pm 0.75}$	$30.81_{\pm 0.94}$	$\underline{33.15}_{\pm 2.87}$
	$^{\sharp}$ICL [75]	$\underline{43.73}_{\pm 1.55}$	$\underline{31.67}_{\pm 5.49}$	$\underline{43.10}_{\pm 3.63}$	$\underline{29.54}_{\pm 4.16}$	$\underline{31.23}_{\pm 0.94}$	$21.91_{\pm 2.35}$
	SPARE (Ours)	$\mathbf{47.51}_{\pm 1.30}$	$\mathbf{43.76}_{\pm 3.14}$	$\mathbf{44.11}_{\pm 1.30}$	$\mathbf{30.72}_{\pm 1.42}$	$\mathbf{35.43}_{\pm 1.10}$	$\mathbf{35.53}_{\pm 2.07}$
Steer to Use Contextual Knowledge							
EM_C	*Without Controlling*	$42.69_{\pm 8.40}$	$41.67_{\pm 4.66}$	$45.96_{\pm 2.48}$	$69.36_{\pm 3.57}$	$62.38_{\pm 3.05}$	$59.25_{\pm 2.82}$
	TaskVec [248]	$41.88_{\pm 9.45}$	$38.25_{\pm 1.23}$	$45.52_{\pm 1.06}$	$\underline{88.47}_{\pm 1.93}$	$\underline{86.91}_{\pm 0.44}$	$59.25_{\pm 1.49}$
	ActAdd [580]	$51.91_{\pm 8.03}$	$47.48_{\pm 3.93}$	$46.90_{\pm 1.89}$	$73.01_{\pm 1.58}$	$69.64_{\pm 0.20}$	$59.66_{\pm 2.89}$
	SEA_{linear} [481]	$43.61_{\pm 10.3}$	$47.73_{\pm 0.43}$	$52.95_{\pm 1.90}$	$69.78_{\pm 0.97}$	$67.32_{\pm 0.28}$	$60.31_{\pm 2.25}$
	SEA_{SqExp} [481]	$57.08_{\pm 2.92}$	$48.04_{\pm 0.45}$	$61.45_{\pm 0.54}$	$72.04_{\pm 1.60}$	$68.20_{\pm 1.10}$	$61.45_{\pm 0.30}$
	DoLa [111]	$44.29_{\pm 8.46}$	$33.54_{\pm 3.38}$	$15.90_{\pm 10.1}$	$68.45_{\pm 3.83}$	$50.95_{\pm 5.15}$	$23.34_{\pm 10.5}$
	$^{\flat}$CAD [535]	$65.65_{\pm 5.50}$	$54.69_{\pm 3.25}$	$63.10_{\pm 2.32}$	$78.69_{\pm 3.85}$	$70.07_{\pm 3.77}$	$\underline{64.12}_{\pm 4.44}$
	$^{\sharp}$ICL [75]	$\underline{73.35}_{\pm 3.82}$	$\underline{63.33}_{\pm 3.50}$	$\underline{70.19}_{\pm 2.51}$	$51.75_{\pm 5.60}$	$47.51_{\pm 1.86}$	$47.24_{\pm 3.81}$
	SPARE (Ours)	$\mathbf{77.69}_{\pm 1.24}$	$\mathbf{69.32}_{\pm 1.26}$	$\mathbf{73.78}_{\pm 0.74}$	$\mathbf{92.24}_{\pm 0.49}$	$\mathbf{87.30}_{\pm 1.96}$	$\mathbf{87.96}_{\pm 1.85}$

Table 5.1. Overall performance of steering the utilisation of parametric and contextual knowledge, measured by EM_M and EM_C. "Without Controlling" indicates the baseline that we do not use any controlling methods to steer the generation. $^{\sharp}$ICL is not an inference-time controlling strategy, which controls the behaviours by changing demonstrations. $^{\flat}$CAD needs additional forwarding for contrastive decoding.

Overall Performance Comparison

Metrics We use Exact Match (EM) to evaluate the performance. Specifically, we evaluate the control capability of generating contextual or parametric answers using the following metrics:

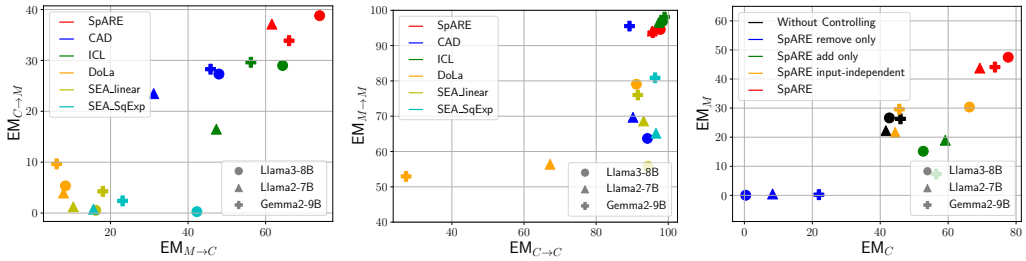
EM_C accuracy of steering the usage of contextual knowledge to generate answers C.

EM_M accuracy of steering the usage of parametric knowledge to generate answer M.

Experimental Results We present the main results in Table 5.1. First, we find SPARE *outperforms existing representation engineering methods* TaskVec, ActAdd and SEA on steering the usage of both contextual and parametric knowledge. This indicates that SPARE can more accurately extract features related to knowledge selection behaviours through the SAE and use them to steer the generation more effectively.

Second, we find SPARE *outperforms contrastive decoding methods* DoLa and CAD, especially in steering the usage of parametric knowledge. Though contrastive decoding strategies can effectively improve the use of contextual knowledge, they struggle to steer the use of parametric knowledge. In contrast, SPARE can more effectively steer the usage of both knowledge by adding and removing the desired and undesired functional features, which we will further analyse in the later ablation study.

Moreover, SPARE *surpasses the non-inference-time controlling method* ICL. It suggests the SPARE can both effectively and efficiently control the knowledge selection behaviours



(a) Changing behaviours. **(b)** Impact of intervention. **(c)** Ablation studies.

Figure 5.4. Detailed evaluation results of controlling capability on NQSwap. We use different colours for different methods and use different shapes for different models. The upper-right area indicates a high performance for all figures. (a) presents the capability of changing the behaviour of LLMs, where x-axis and y-axis are $\text{EM}_{C\to M}$ and $\text{EM}_{M\to C}$, measuring the capability of changing the answer from C to M and from M to C, respectively; (b) presents the capability of maintaining the behaviour when steering to the same behaviour as the original behaviour, where x-axis and y-axis are $\text{EM}_{M\to M}$ and $\text{EM}_{C\to C}$, measuring the maintaining capability of generating M and C, respectively; (c) present the ablation analysis of SpARE, x-axis and y-axis are EM_M and EM_C.

of LLMs. It also suggests a promising capability of representation engineering to control the behaviours of LLMs at inference time in practical applications.

Multi-Perspective Controlling Analysis

In the following, we analyse the controlling capability of SpARE from different perspectives: *1)* the capability of changing the behaviour (Fig. 5.4a), *2)* the potential negative impact of intervention (Fig. 5.4b), and *3)* the ablation study (Fig. 5.4c).

Capability of Changing the Behaviours Unlike merely comparing overall performance across the entire dataset, we further examine their capability of changing the original knowledge selection behaviour of LLMs by the following two metrics:

$\text{EM}_{C\to M}$ accuracy of changing the behaviour from generating contextual answers C to parametric answers M in the subset of instances where the model generates C without controlling.

$\text{EM}_{M\to C}$ accuracy of changing the behaviour from generating parametric answers M to contextual answers C in the subset of instances where the model generates M without controlling.

As shown in Fig. 5.4a, SpARE outperforms contrastive decoding methods and is located in the upper-right area of the figure. SpARE also outperforms representation engineering methods. It suggests that the SAE enables accurately extracting features related to knowledge behaviour and thus more effectively changes both the original behaviours of using contextual and parametric knowledge. We also observe that all methods are less effective in steering toward the use of parametric knowledge than contextual knowledge. This finding matches the previous works [554, 626, 442], which shows LLMs prefer contextual knowledge, and thus more difficult to steer the behaviour of using parametric knowledge.

Impacts of Intervention A sufficient but unnecessary intervention can change the behaviour of LLMs, but it also can introduce noise and decrease accuracy. Here, we investigate the potential negative impact of methods by steering LLMs using the same knowledge they will use without control. We expect LLMs to maintain their original behaviours, measured by the following two metrics:

$\text{EM}_{M\to M}$ accuracy of maintaining the behaviour of generating M when steering the use of parametric knowledge in the subset of instances where the model generates M without controlling.

$\text{EM}_{C\to C}$ accuracy of maintaining the behaviour of generating C when steering the use contextual knowledge in the subset of instances where the model generates C without controlling.

As shown in Fig. 5.4b, as it minimally alters the original behaviour when guiding the model to produce similar outcomes. SPARE has a close performance to ICL, indicating it can steer the behaviour effectively while introducing a little unnecessary editing. Though CAD maintains the most accuracy in contextual knowledge, its performance decreases substantially in maintaining the behaviour of generating parametric knowledge. Finally, while other representation engineering methods may alter the entire model behaviour due to editing of polysemantic features, SPARE provides a more precise approach to editing through the SAE activations and thus delivers better performance in maintaining the model behaviours.

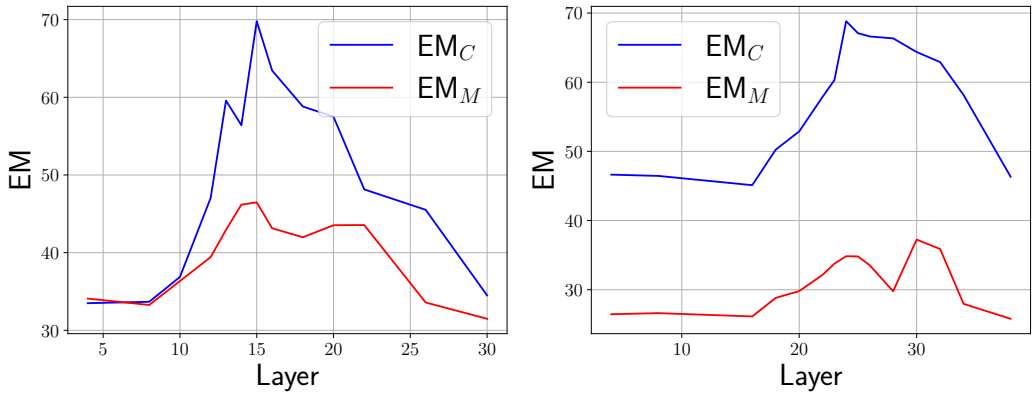
Ablation Study We present the ablation study in the following settings: *1)* SPARE input-independent: it uses $\mathbf{z}_C$ and $\mathbf{z}_M$ to steer the generation without calculating $\mathbf{z}^-$ and $\mathbf{z}^+$ based on the input activation; *2)* SPARE remove only: it edits the hidden states by only removing the functional features of the undesired behaviour; *3)* SPARE add only: it edits the hidden states by only adding the functional features of the desired behaviour.

As shown in Fig. 5.4c, we can see that every ablation results in a significant controlling capability decrease. The input-independent editing strategy that omits the calculations of Eq. (5.2) and Eq. (5.3) fails to steer the usage of knowledge and obtain results that are close to the ones we obtain without controlling. The results of SPARE "remove only" obtain a zero accuracy on both EM_M and EM_C, indicating that the model cannot keep the original behaviour and also cannot generate answers toward another behaviour. This suggests that SPARE can effectively remove the functional features of the original behaviour. Moreover, SPARE "add only" leads to worse performance than without controlling, suggesting the importance of removing the features of undesired knowledge selection behaviour.

5.1.5 Analysis and Discussion

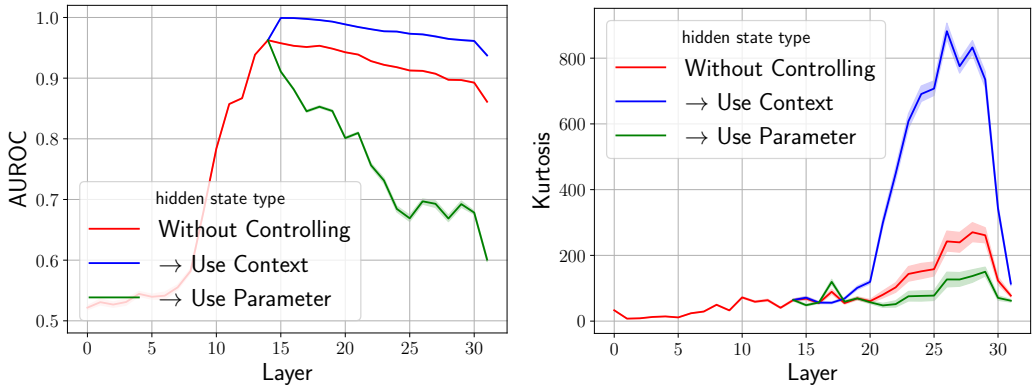
Analysing the Layer Choice

We present the results of editing multiple layers in Table 5.1; here, we analyse the effectiveness of SPARE by editing each layer individually. As shown in Fig. 5.5, we find SPARE can control the behaviour of LLMs most effectively at mid-layers for both Llama3-8B and Gemma2-9B. These layers are also where we can detect knowledge conflict most



(a) Llama3-8B **(b)** Gemma2-9B

Figure 5.5. Effectiveness of SPARE on editing different layers individually.



(a) Probing results **(b)** Skewness pattern

Figure 5.6. The residual stream changes after applying SPARE to Llama3-8B at the 15th layer.

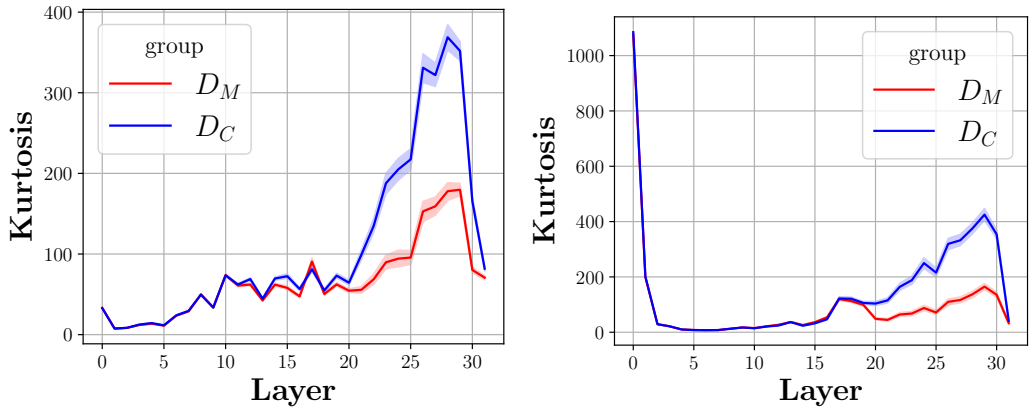
accurately, as shown in Fig. 5.2 and Section 5.1.7. This supports the practical application of inference-time intervention to control knowledge selection behaviour, where SPARE can effectively steer the generation once we detect the conflict.

The effectiveness of steering the behaviours in middle layers also matches previous findings [248, 446, 570], that suggest that the middle layers of LLMs contain the functional feature that drives a task. To the best of our knowledge, we are the first to accurately extract this functional feature using pre-trained SAEs.

Analysing the Residual Stream

We analyse how the residual stream changes after applying SPARE. Here, we edit the hidden states from $\mathcal{D}_{E_C} = \{(Q, E_C)\}$ at the 15th layer to steer the contextual and parametric knowledge usage.

In Fig. 5.6a, we present the probing results on the residual stream using the same probing model described in Section 5.1.2. We observe that when we steer towards the usage of parametric knowledge, the probing performance decreases *immediately*



(a) Llama3-8B **(b)** Llama2-7B

Figure 5.7. The skewness patterns of the residual steam when LLMs select different sources of knowledge to generate the answer without controlling in NQSwap.

(green line), indicating that the signal of knowledge conflict fades quickly, and the representations of activations become closer to $\mathcal{D}_{E_M}$, thus making it more difficult for the probing model to distinguish whether a given activation is from $\mathcal{D}_{E_C}$ or $\mathcal{D}_{E_M}$. In contrast, when we steer towards using contextual knowledge, the probing performance increases (blue line), indicating the signal of the conflict increases, and the representations of activations become more different from $\mathcal{D}_{E_M}$, making it easier for the probing model to distinguish whether a given activations is from $\mathcal{D}_{E_C}$ or $\mathcal{D}_{E_M}$.

In Fig. 5.6b, we find the skewness of the representation from the residual stream – measured by Kurtosis – shows distinct patterns after applying SPARE. We observe that when we apply SPARE to steer the usage of contextual knowledge at the 15th layer, the residual stream becomes significantly more skewed starting from *later layers*—the 19th layer (blue line); in contrast, when we use parametric knowledge, the residual stream becomes less skewed (green line). Moreover, in Fig. 5.7, we analyse the skewness pattern when LLMs freely select knowledge to generate answers without controlling. We find the residual stream of $\mathcal{D}_C$, where the model uses contextual knowledge to generate answers, is significantly more skewed than $\mathcal{D}_M$ from the 19th layer. Thus, the skewness pattern changes shown in Fig. 5.6b can indicate that SPARE steers the knowledge selection behaviours.

We provide more analysis of the representation patterns in Section 5.1.12 and [679]. In this work, we only provide our empirical observation on the representation pattern and leave investigating the reasons in future works.

5.1.6 Related Work

Representation Engineering Many studies focus on *mechanistic interpretability* to understand the LLMs by analysing the activities and connections of individual network components, such as circuits [176, 435] and neurons [203, 393] However, though mechanistic interpretability can successfully explain simple mechanisms, it often struggles with more complex phenomena [699]. Differently, *representation engineering* [580, 481, 699] offers a complementary approach. It focuses on the characteristics of representations

rather than lower-level mechanisms, providing a framework for understanding complex systems at a higher level of abstraction. It has shown more promise in interpreting higher-level behaviours of LLMs at scale. Some works modify model activations to change behaviours [496, 274, 367, 700, 339], and some works extract latent vectors and leveraging these vectors to regulate the model's inference [581, 555, 500].

Knowledge Conflicts *Knowledge conflicts* refer to discrepancies among contextual and parametric knowledge [95, 627]. [633] identify three types of knowledge conflicts: *inter-context* [667, 169, 447, 681], *context-memory* [372, 627, 404], and *intra-memory conflicts* [268]. In this work, we focus on context-memory knowledge conflict, which refers to conflicts between the contextual knowledge and the parametric knowledge encoded in the model parameters. [442] and [286] investigate the mechanisms of attention heads and feed-forward networks of LLMs when context-memory knowledge conflict occurs.

Sparse Auto-Encoder Sparse Auto-Encoders (SAEs) have been introduced as a post-hoc analysis tool to identify disentangled features within uncompressed representations of an LLM [659, 74, 271]. SAEs are trained with sparsity regularisation to learn a sparse, overcomplete basis that characterises the activation space of an LLM [50]. [383] showed that the features learned by SAEs can identify sparse circuits in LLMs. [567] showed the possibility of searching for monosemantic features and steering LLMs' generation. [91] improves steering vectors using SAEs. [234] generates description for representations learned through SEAs. [321] finds universal representation across different language models.

Limitations

While our proposed method, SPARE, demonstrates effective control over knowledge selection behaviours in LLMs, there are several limitations to consider. First, the approach relies on pre-trained SAEs to identify and manipulate functional features within the internal activations of the model, so it may not directly apply to models where pre-trained SAEs are unavailable or cannot be efficiently trained. Second, our experiments are conducted on specific ODQA tasks involving context-memory knowledge conflicts. While the results are promising in this setting, it is unclear how well the method generalises to other types of tasks or conflicts, such as those involving complex reasoning, multi-hop questions, or long-form generation. Finally, the control over knowledge selection behaviours is evaluated primarily in terms of steering the model towards either contextual or parametric knowledge. In practice, the decision about which knowledge source to trust may not be binary or require a more elaborate approach, such as using another model as a critic [255].

5.1.7 More Analysis of Knowledge Conflict Probing

Details of Probing Model

We train the probing model with an L_1 norm regularisation for all probing experiments. The training objective is $\mathcal{L} = -\log P(y = y_i) + \lambda \|W\|_1$, where we set λ to 3×10^{-4} and

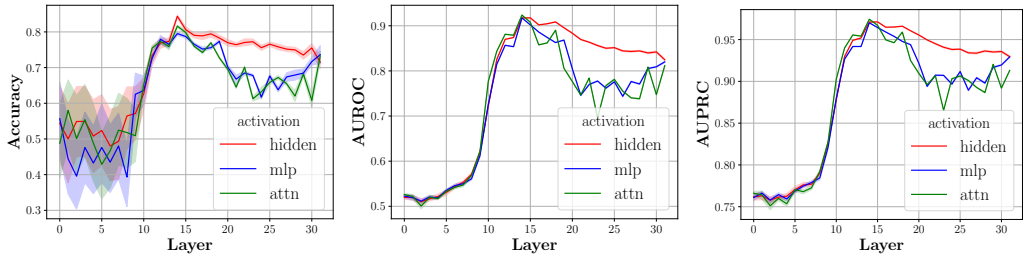


Figure 5.8. Knowledge conflict probing results using Llama2-7B on NQSwap.

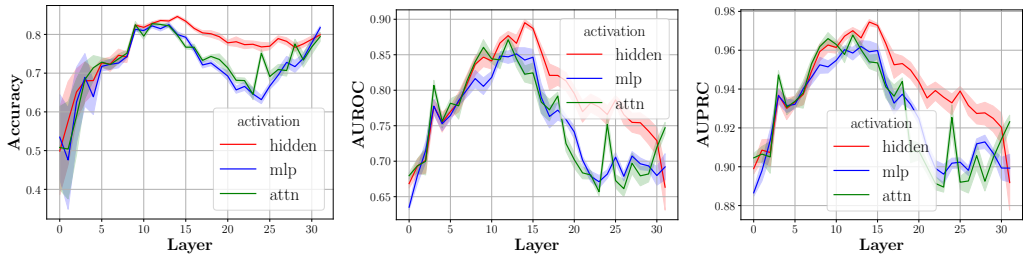


Figure 5.9. Knowledge conflict probing results using Llama2-7B on Macnoise.

y_i is the label. We train 20 times with different random seeds for each probing task, and we report the average and deviation in our experiments. We split the training and test datasets for the probing tasks, ensuring no overlapping questions between them.

More Probing Results

We present the knowledge conflict probing results of Llama2-7B on NQSwap and Macnoise on Fig. 5.8 and Fig. 5.9 using accuracy, AUROC and AUPRC as metrics. We provide more analysis of probing the residual stream under knowledge conflict in our preliminary study [679].

5.1.8 Sparse Auto-Encoders Details

Both the SAEs of Llama3-8B from ElethuerAI and Llama2-7B pre-trained by us have a latent dimension $n = 131072$, 32 times larger than the number of dimensions of the hidden state size $d = 4096$, and use ReLU as activation function σ (Eq. (5.1)). Gemma2-9B uses JumpReLU [491] as the activation function. GemmaScope [353] provides different sizes of SAEs, and we use the size of $n = 131072$ in the experiment.

Following [194], we pre-train the SAEs for Llama2-7B with TopK activation functions: $\mathbf{z} = \text{TopK}(\mathbf{W}_\theta(\mathbf{h} - \mathbf{b}))$, which only keeps the k largest latents during pre-training, while does not use sparsity regularisation during pre-training. The loss is calculated by the L_2 norm of reconstruction error: $\mathcal{L} = \|\mathbf{h} - \hat{\mathbf{h}}\|_2^2$. We pre-train SAE models[2] with 10B tokens sampled from RedPajama[3] and use the hyperparamters determined by [194]. The pre-training for an SAE for a certain layer of hidden states costs about 300 80G

[2] https://github.com/EleutherAI/sae
[3] https://huggingface.co/datasets/togethercomputer/RedPajama-Data-V2

A100 GPU hours.[4]

Though it costs non-trivial resources for pre-training SAEs, we believe it is valuable to explore using SAEs to resolve knowledge conflicts for the following reasons: *1)* SAEs are general models for interpreting the representation of LLMs, which have broader applications beyond steering knowledge selection behaviours; *2)* SAEs are becoming popular tools for interpreting LLMs with rising numbers of open-resource frameworks and pre-trained models released recently, so we can use the pre-trained SAEs conveniently rather than pre-training SAEs by ourselves in the future.

5.1.9 Implementation Details

Collecting Activations

Collecting activations is an essential step in representation engineering methods [699, 481], where we will extract desired features from the collected activations and use these features to edit the activations of LLMs. In SPARE, we first sample demonstrations from the development set using 5 seeds. Then, we use these demonstrations to test the rest of the questions with conflict evidence E_C. Based on the predictions, we split the instance into $\mathcal{D}_C$ and $\mathcal{D}_M$ and collect their corresponding hidden states, denoting as $\{\mathbf{h}_C^j\}_{j=1}^N$ and $\{\mathbf{h}_M^j\}_{j=1}^N$. Then, the corresponding layer's SAE encodes them to $\{\mathbf{z}_C^j\}_{j=1}^N$ and $\{\mathbf{z}_M^j\}_{j=1}^N$. Here, we use j to index the collected instances and use i to index the SAE's activation.

One strategy to highlight the values of behaviour-related activation is weighted averaging $\{\mathbf{z}_C^j\}_{j=1}^N$ and $\{\mathbf{z}_M^j\}_{j=1}^N$ by the confidence of generating a specific answer. Specifically, denote the confidence of generating answers C and M conditioned on $\mathbf{h}_C^j$ and $\mathbf{h}_M^j$ as

$$\lambda_C^j = \frac{\log P(C \mid \mathbf{h}_C^j)}{\log P(C \mid \mathbf{h}_C^j) + \log P(M \mid \mathbf{h}_C^j)}$$
$$\mu_M^j = \frac{\log P(M \mid \mathbf{h}_M^j)}{\log P(C \mid \mathbf{h}_M^j) + \log P(M \mid \mathbf{h}_M^j)},$$

where $P(\cdot \mid \mathbf{h})$ is calculated by the output of LLMs. We use the normalised confidence $\lambda_C^{'j}$ and $\mu_M^{'j}$ as weight to average $\{\mathbf{z}_C^j\}_{j=1}^N$ and $\{\mathbf{z}_M^j\}_{j=1}^N$, respectively:

$$\overline{\mathbf{z}_C} = \sum_{j=1}^{N} \lambda_C^{'j} \mathbf{z}_C^j, \text{ and } \overline{\mathbf{z}_M} = \sum_{j=1}^{N} \mu_M^{'j} \mathbf{z}_M^j.$$

We find this strategy brings a slight improvement compared to directly average $\{\mathbf{z}_C^j\}_{j=1}^N$ and $\{\mathbf{z}_M^j\}_{j=1}^N$.

Identifying Functional Activations

We identify the activations that steer the usage of contextual and parametric knowledge by calculating the mutual information between activation Z_i and the generated answer

[4]We provide our pre-trained SAEs in https://huggingface.co/yuzhaouoe/Llama2-7b-SAE/tree/main

$Y = \{C, M\}$:

$$I(Z_i; Y) = \sum_{z_i \in Z_i} \sum_{y \in \{C,M\}} P(z_i, y) \log \frac{P(z_i, y)}{P(z_i)P(y)}.$$

Since a higher $I(Z_i; Y)$[5] indicates a higher dependency between Z_i and the knowledge selection behaviour, we sorted $\{I(Z_i; Y)\}_{i=1}^{n}$ in descending order, and consider the top k activation work as functional activations. To decide the number of selected activations k, we introduce a hyperparameter K. We then choose k such that:

$$k = \arg\min_{k} \sum_{i=1}^{k} \frac{I(Z_i; Y)}{\sum_{j=1}^{n} I(Z_j; Y)} \geq K, \tag{5.5}$$

which means selecting the top k SAE activations with the proportion K% in the sum of all mutual information $\sum_{j=1}^{n} I(Z_j; Y)$. One potential improvement is normalising mutual information based on entropy, which has shown better properties for comparing the importance of features. We determine the top activations in each layer individually rather than ranking all mutual information across layers.

The proportion K abstracts the exact number of activations to select. We expect the same types of SAEs, e.g., pre-trained by TopK [194] or JumpReLU [491], can share similar values of K for controlling. We present the relation between k and K in Section 5.1.11. We also present the selected Gemma2-9B SAEs activations used by SpARE in Section 5.1.10.

Impact of the Size of the Collected Activations

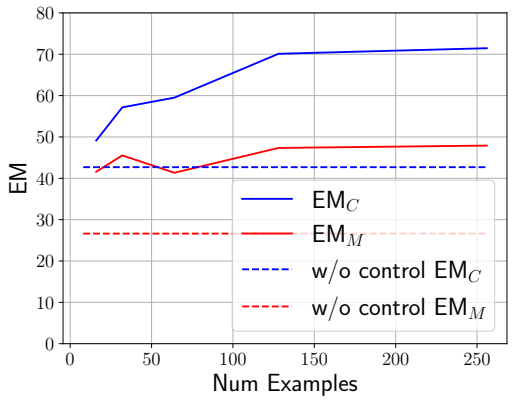
We collect the hidden states $\{\mathbf{h}_C^j\}_{j=1}^{N}$ and $\{\mathbf{h}_M^j\}_{j=1}^{N}$ to calculate the value of functional activations $\mathbf{z}_C$ and $\mathbf{z}_M$, and we also use $\{\mathbf{z}_C^j\}_{j=1}^{N}$ and $\{\mathbf{z}_M^j\}_{j=1}^{N}$ to estimate the mutual information. Here, we analyse the impact of N on the controlling performance.

As shown in Fig. 5.10a, the performance increases when we use 8 examples to 128 for calculating the mutual information, while collecting more activations brings slight improvement. In Fig. 5.10b, the performance does not increase until using 64 examples to calculate $\mathbf{z}_C$ and $\mathbf{z}_M$. The above analysis indicates that SpARE needs at least 128 activations to achieve a high controlling capability.

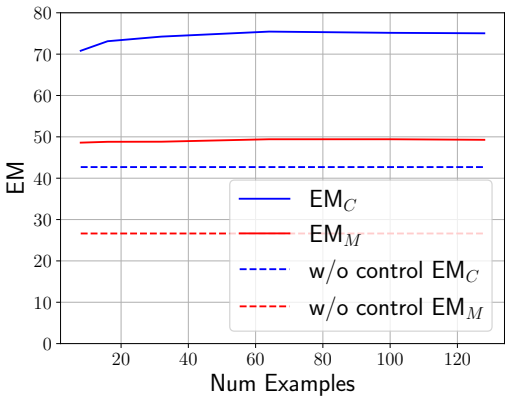
Development Set and Demonstrations

We held out a demonstration set consisting of 128 instances from each dataset for each model. Each question in the demonstration set can be answered correctly by the corresponding model without providing evidence; more specifically, we test it with the close-book setting with few-shot examples to align the answer format. We use the non-conflict context E_M and memorised answer M in demonstrations. Since the contextual and parametric knowledge are consistent in the demonstration set, they do not provide information about how to resolve the knowledge conflict for the test examples when presented with conflict evidence E_C. We sample 5 different sets of demonstrations

[5] We estimate mutual information between continuous variables Z_i and discrete labels Y using https://scikit-learn.org/1.5/modules/generated/sklearn.feature_selection.mutual_info_classif.html



(a) Use different numbers of activations to calculate mutual information. (Section 5.1.3)



(b) Use different numbers to calculate $\mathbf{z}_C$ and $\mathbf{z}_M$. (Section 5.1.3)

Figure 5.10. The impact of the number of the collected hidden states N on the controlling performance.

using 5 different seeds and present the average results with deviations. All methods we compared use the same sets of demonstrations for each model.

The held-out demonstration set is also used as the development set to search hyper-parameters for each model in each dataset. We sample a set of demonstrations to align the answer format and use the rest of the instances to evaluate the performance.

Implementation Details of Representation Engineering Baselines

For all representation engineering baselines TaskVec [248], ActAdd [580] and SEA [481], we experimented with performing the intervention at different layers and reported the best performance in each case. We present the implementation details as follows.

TaskVec [248]: We first collect the task vectors using a few-shot setup where the few-shot examples contain the conflict evidence E_C along with the memorised answer M. We then follow the procedure illustrated in the original paper and extract the Task Vectors as the hidden representation of the last token (: in our case) at all layers. More specifically, we use the hidden representation of samples that generate answers following the context to create $\overline{\mathbf{h}_C}$, while we use those that follow the parametric knowledge and ignore the context to create $\overline{\mathbf{h}_M}$. At inference time, we replace the residual stream at layer L with $\overline{\mathbf{h}_C}$ or $\overline{\mathbf{h}_M}$ to steer the model to follow the context or the parametric knowledge, respectively.

ActAdd [580]: We collect the activations for ActAdd following the same procedure. For the inference-time intervention, we edit the residual stream by adding $\overline{\mathbf{h}_C}$ and subtracting $\overline{\mathbf{h}_M}$ to steer the model towards context knowledge, while we perform the opposite to steer the model towards parametric knowledge.

SEA [481]: SEA adopts a different strategy and uses positive, negative and neutral model generations to compute steering vectors. We consider as neutral the generations that result from a few-shot setup where the demonstrations contain the memorised evidence E_M and memorised answer M, irrespective of the generated output. We then collect activations $\overline{\mathbf{h}_M}$ and $\overline{\mathbf{h}_C}$ following the same procedure illustrated above and use the method described in [481] to compute the steering vectors, assuming $\overline{\mathbf{h}_M}$ and $\overline{\mathbf{h}_C}$ encode the positive (follow parametric knowledge) and negative (follow context knowledge) behaviours respectively.

Searching Hyperparameters

We search hyperparameters of all methods using the same set of development sets. We choose hyperparameters based on the $\text{EM}_{C\to M}$ and $\text{EM}_{M\to C}$, which measure the capability of changing the behaviours of LLMs.

DoLa [111]: We search the best coefficient α that is used to compare premature and mature logits, evaluating with the "higher-layer" and "lower-layer" settings:

$$\log P(y) = \log P_{\text{mature}}(y) - \alpha \log P_{\text{premature}}(y).$$

We test the α ranging from -10 to 10 with an interval of 0.5. Finally, we set $\alpha = 6.0$ and $\alpha = -8.0$ to steer the model to use contextual and parametric knowledge for Llama2-7B and Llama3-8B; set $\alpha = 1.0$ and $\alpha = -1.0$ for Gemma2-9B. However, based on our experiments, though DoLa has a certain ability to change the behaviours of Gemma2-9B, we do not find a suitable α on both "high-layer" and "low-layer" choices for improving Gemma2-9B on the overall performance measured by EM_M and EM_C.

CAD [535]: We search the best coefficient α that is used to combine the logits with context and the logits without the context:

$$\log P(y) = (1+\alpha)\log P(y \mid c, x) - \alpha \log P(y \mid x).$$

We test the α ranging from -1.5 to 1.5 with an interval of 0.1. Finally, we set $\alpha = 0.6$ and $\alpha = -0.8$ to steer the model to use contextual and parametric knowledge for Llama2-7B and Llama3-8B; set $\alpha = 0.3$ and $\alpha = -0.3$ for Gemma-2-9B.

SPARE (Ours): Since no previous work used SAEs to control the knowledge selection behaviour of LLMs, we need first to identify suitable magnitudes of the hyperparameters and then search them in a smaller range in the development set. We fix the $\alpha = 1$ in Eq. (5.4) and test the proportion K to select the activations ranging from 0.1 to 0.01 with an interval of 0.01. Then, we choose $K = 0.07$ and test α ranging from 1.0 to 3.0 with an interval of 0.2. Finally, we select $K = 0.07$ and $\alpha = 2$ for Llama3-8B, $K = 0.06$ and $\alpha = 2.2$ for Llama2-7B. In our main experiments, SPARE edits the 13th to the 16th layers of Llama3-8B and the 12th to the 15th layers of Llama2-7B. We do not try other choices because there is no public SAE for Llama2-7B.

Due to the Gemma2-7B's SAEs [353, 491] using different training strategies and activation functions, they show a much more sparse pattern and have different suitable

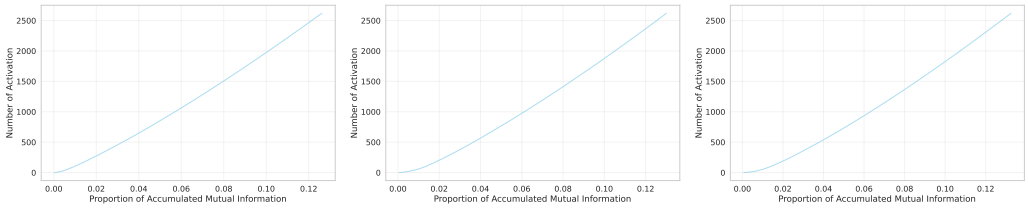
hyperparameters. Here, we select $K = 0.01$ and $\alpha = 3$ to steer contextual knowledge, and $K = 0.01$ and $\alpha = 1.8$ to steer parametric knowledge. In our main experiments, SPARE edits 23, 24, 25, 29, 30 and 31 layers of Gemma2-9B.

We also present the selected top k activations in Section 5.1.10 for further analysis.

5.1.10 Selected SAEs Activations of Gemma2-9B

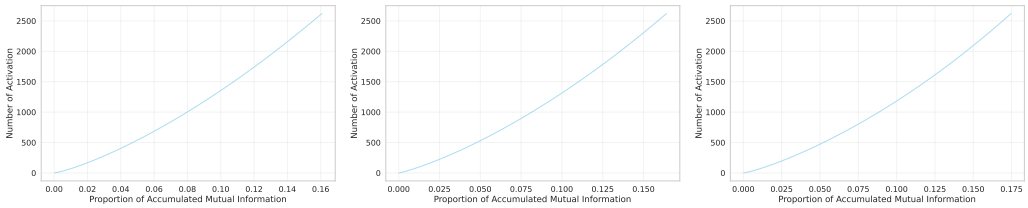
In Table 5.2, we present the selected SAE activations used by SPARE for steering the knowledge selection behaviour. We can further interpret them in GemmaScope[6] [353].

[6] https://www.neuronpedia.org/gemma-scope



(a) Layer 23 **(b)** Layer 24 **(c)** Layer 25

Figure 5.11. Proportion of accumulated mutual Information (K) on Gemma2-9B



(a) Layer 13 **(b)** Layer 14 **(c)** Layer 15

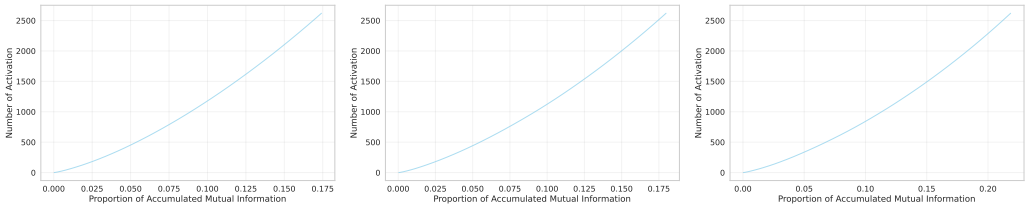
Figure 5.12. Proportion of accumulated mutual Information (K) on Llama2-7B

5.1.11 Distribution of Mutual Information

In Section 5.1.9, we mentioned that we use the proportion mutual information (K) to determine how many activations of the SAE (k) to select. Figs. 5.11 to 5.13 shows the layer-wise accumulated mutual information (x-axis) for the number of selected activations (y-axis) across different models (Gemma2-9B, Llama2-7B, Llama3-8B). In all three models, we observed that the graph is skewed when k (y-axis) is small, indicating that some SAE activations have relatively high mutual information. While there were some differences in this tendency between models (particularly pronounced in Gemma2-9B), we found only a little variation across selected layers within the same model. This analysis corresponds to the K values (from 0.01 for Gemma2-9B to 0.07 for Llama3-8B) that we identified through hyperparameter search in Section 5.1.9.

5.1.12 Distribution Patterns of the Residual Stream Under Knowledge Conflict

In this section, we provide further analysis of the representation patterns when knowledge conflicts in addition to Fig. 5.6b. Here, we focus on analysing the distribution patterns of different knowledge selection behaviours. More specifically, we compare the representation difference between the activation from $\mathcal{D}_C$ and $\mathcal{D}_M$, which are both under knowledge conflict but select contextual and parametric knowledge to generate the answer C and M. In Section 5.1.12, we analyse the skewness patterns; in Section 5.1.12, we analyse the L1 norm and L2 norm patterns since previous work [159] also show the norm value may be related to the contextual information usage. We provide more residual stream analysis in our preliminary study [679].



(a) Layer 13 **(b)** Layer 14 **(c)** Layer 15

Figure 5.13. Proportion of accumulated mutual Information (K) on Llama3-8B

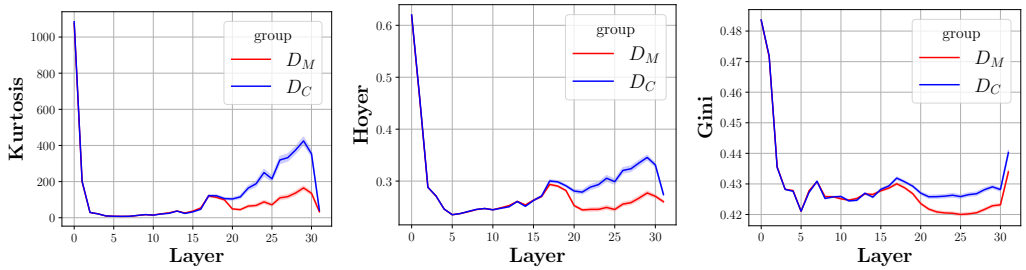


Figure 5.14. Skewness of the hidden states of Llama2-7B on NQSwap.

Skewness of Residual Stream

In addition to Kurtosis, we used in Fig. 5.6b, we also measure the skewness by Hoyer and Gini index. We present the skewness patterns of hidden states in Fig. 5.14 and Fig. 5.15. We find the residual stream exhibits a significant skewed pattern when selecting the contextual knowledge to generate the answer. This observation supports the effectiveness of SPARE, where the residual steam becomes skewed when SPARE steers the model to generate contextual knowledge as shown in Fig. 5.6b.

We also analyse the skewness pattern of MLP activations and Self-Attention activations in Fig. 5.16 and Fig. 5.17. However, we do not observe a distinct distribution difference like hidden states.

L1 Norm and L2 Norm Pattern

As we observe the distinct skewness pattern in hidden states, we further analyse their L1 norm and L2 norm patterns in Fig. 5.18 and Fig. 5.19 However, we do not observe the distinct norm differences between $\mathcal{D}_C$ and $\mathcal{D}_M$, though they have a significantly different skewness pattern.

5.1.13 Conclusion

We investigated the context-memory knowledge conflicts in LLMs. We identify that knowledge conflicts can be detected by probing the residual stream of the model (Section 5.1.2) and propose SPARE (Section 5.1.3), a training-free representation engineering method that leverages pre-trained SAEs to effectively and efficiently control the knowledge selection behaviour of LLMs at inference time. Our experimental results on ODQA tasks show that SPARE produces more accurate results than existing representation engi-

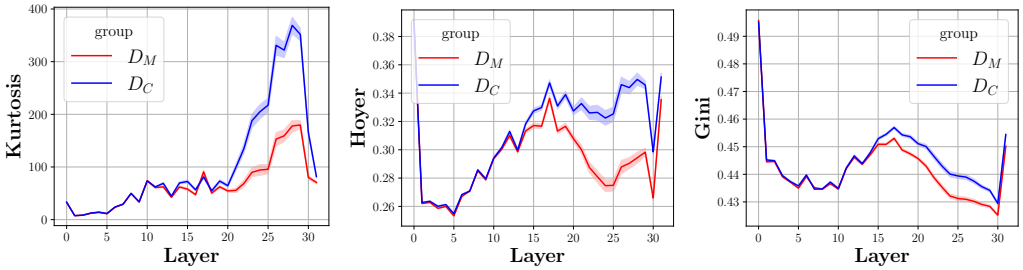


Figure 5.15. Skewness of the hidden states of Llama3-8B on NQSwap.

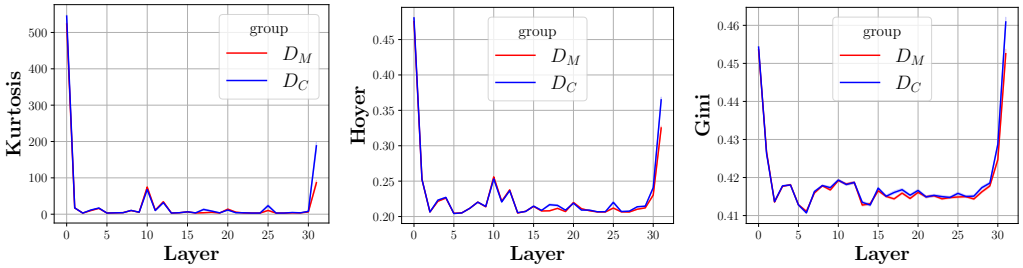


Figure 5.16. Skewness of the MLP activation of Llama2-7B on NQSwap.

neering and contrastive decoding methods (Section 5.1.4). By providing a mechanism to steer knowledge selection behaviours at inference time, SPARE offers a promising approach to managing knowledge conflicts in LLMs without significant computational overhead. Additionally, we investigate the residual stream of LLMs under knowledge conflicts (Section 5.1.5). We find that *1)* the knowledge selection behaviour is more steerable at the middle layers of LLMs; *2)* the residual stream shows a significantly more skewed representation when models using contextual knowledge compared to using parametric knowledge.

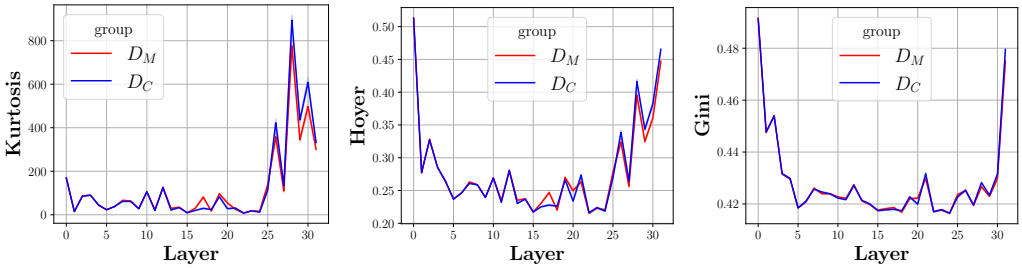


Figure 5.17. Skewness of the Self-Attention activation of Llama3-8B on NQSwap.

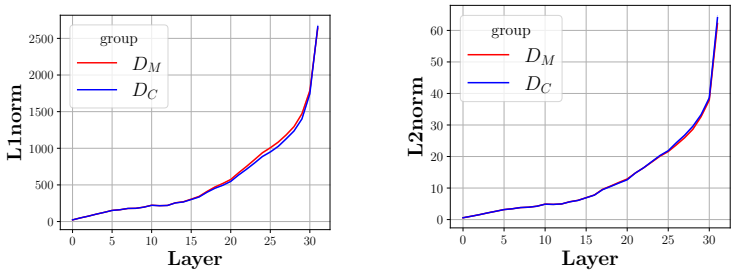


Figure 5.18. L1 norm and L2 norm of the hidden states of Llama3-8B on NQSwap.

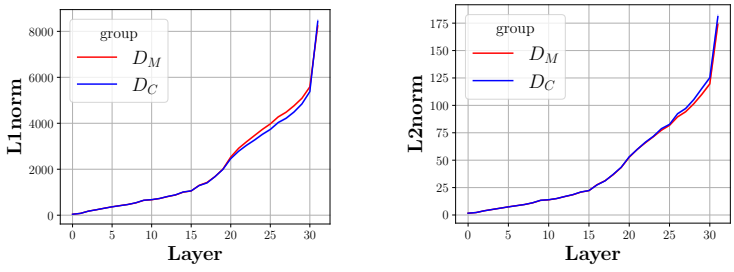


Figure 5.19. L1 norm and L2 norm of the hidden states of Llama2-7B on NQSwap.

Layer	Use Parametric Knowledge	Use Contextual Knowledge
23	116391, 36331, 85142, 2795, 99547, 63615, 25635, 123378, 105328, 24132, 113025, 83008, 37706, 60782, 36046, 110864, 101469, 29902, 129485, 112858, 104185, 17911, 6673, 72533, 108414, 32967, 19761, 118260, 109917, 55083, 41965, 91874, 74605, 19726, 115338, 80100, 3042, 48088, 61830, 895, 49288, 120379, 105552, 84782, 14129	59646, 66244, 130943, 100165, 103568, 82090, 116937, 108558, 78302, 100628, 53091, 90600, 124049, 63656, 118525, 119623, 34458, 119574, 38170, 66293, 14026, 28797, 125520, 76467, 29583, 89951, 32901, 52256, 130987, 36816, 59062, 58505, 123631, 60183, 11432, 86969, 11755, 71200, 53746, 33, 57883, 67097, 108617, 112319, 1380, 47638, 42621, 16859, 130470, 6475, 112033, 101316, 40945, 82574, 58929, 79660, 81043, 18549, 4537, 130935, 127945, 78809
24	10649, 68997, 80242, 38885, 33450, 29004, 34725, 55203, 41474, 90933, 118013, 76436, 2795, 53138, 41501, 65408, 116855, 12056	76071, 55422, 82954, 40832, 68001, 88619, 120959, 92931, 38262, 83042, 42129, 21413, 74005, 73350, 57270, 6859, 83385, 9263, 8609, 22968, 8307, 99263, 2415, 59807, 87788, 92845, 88733, 124321, 25758, 111976, 84892, 104309, 61391, 60162, 128726, 28753, 62671, 80398, 40150, 28432, 81514, 9463
25	117145, 66103, 55992, 1609, 101788, 28707, 64494, 63602, 81174, 73438, 16428, 2054, 44642, 12418, 105769, 37692, 33693, 22786	77008, 39999, 65977, 3002, 82187, 113845, 35985, 16341, 121937, 13762, 9468, 70433, 42102, 85578, 3118, 99639, 41828, 58588, 103815, 70243, 67915, 125985, 113290, 127536, 84912, 2473, 46174, 100026, 37216, 27820, 81800, 13540, 125213, 79326, 55733, 32460, 46612
29	70665, 84563, 63717, 45653, 122282, 5001, 67756, 52905, 118450, 84589, 16721, 119640, 47070, 15218, 117432, 110719, 98957, 11667, 20824, 31422, 119807, 22664, 81261, 116958	65636, 113411, 88779, 19501, 46209, 8584, 71156, 79159, 94888, 144, 60280, 413, 103986, 74324, 52419, 70057, 30294, 13647, 37430, 71657, 118541, 12744, 74953, 115544, 19086, 102886, 49216, 95333, 26177, 89774, 71927, 70989, 23760
30	116964, 47548, 20615, 48375, 128786, 1308, 40865, 22211, 15816, 107813, 50419, 113319, 97588, 30688, 110627, 56882, 117785, 63602, 39609, 52155, 99243, 36852, 121514, 73310, 850, 96578	84358, 115174, 11363, 28696, 110664, 2831, 24365, 128820, 35092, 92968, 78722, 22739, 128047, 127030, 77294, 76467, 74131, 56766, 94697, 58000, 32812, 46910, 82749, 106077, 59596, 103936, 4505, 129363, 126847, 42463, 120310
31	61476, 5054, 1364, 18335, 63832, 88313, 35780, 130003, 25371, 125651, 11685, 24947, 2260, 70799, 92415, 47791, 99787, 88517, 85499, 75095, 114075, 125055, 109519, 116785, 100449, 37567, 88965, 59674, 14203, 125588, 70706, 18151	121514, 35148, 15479, 65369, 18623, 98225, 52746, 45804, 107893, 10202, 69463, 83810, 12131, 111417, 115174, 107085, 26328, 75203, 37430, 127639, 18114, 80704, 68360, 33142, 51607, 96802, 24949, 97568, 82042, 50826, 110615, 110929, 97833

Table 5.2. Selected Gemma2-9B SAEs activations with $K = 0.01$ (Eq. (5.5)). "Use Parametric Knowledge" means these activations are positively correlated with the behaviour of selecting parametric knowledge, determined according to our method described in Section 5.1.3.

5.2 Interpretable Mixture of Experts Graph Transformer for Particle Collision Detection

The Large Hadron Collider (LHC) at CERN offers an exceptional setting for investigating the fundamental constituents of matter. Experiments like ATLAS [26], generate extensive data streams from high-energy particle collisions, enabling the study of particle interactions across a wide energy spectrum. While this data is invaluable for advancing particle physics, the sheer scale and complexity of the information present significant analytical challenges [60]. For instance, ATLAS event reconstruction processes data at rates exceeding 3.5 terabytes per second [30], requiring advanced computational techniques to extract meaningful insights.

In this work, we leverage simulated LHC data as a controlled benchmark to address a key challenge in modern high-energy physics (HEP): the development of interpretable machine-learning models. Simulated data, where the underlying physics is fully known, provides an ideal environment to evaluate the trustworthiness of machine-learning methods by verifying whether their outputs align with established physical principles. Machine learning (ML) and deep learning techniques have demonstrated strong potential in tackling HEP challenges [229], including particle identification [27], event reconstruction [546], and background subtraction [120]. Nevertheless, these methods often operate as "black boxes", making it difficult to interpret their decision-making processes. This lack of transparency is a critical limitation, particularly in fields like HEP, where explainability is essential for establishing confidence in novel discoveries.

From a performance point of view, Graph Neural Networks (GNNs) have emerged as promising tools for analyzing complex datasets like those produced by the LHC. By representing particles and their interactions as nodes and edges in a graph [568], GNNs can model intricate relationships that other methods might overlook. While GNNs have demonstrated state-of-the-art performance in classification tasks, their value extends beyond predictive accuracy. A major focus of this work is on developing GNN-based models that inherently provide insights into their internal decision-making processes, ensuring that their predictions align with physical intuition. The terms *explainability* and *interpretability* are often used interchangeably in the literature, although some works attempt to distinguish between them. However, there is currently no universally accepted consensus within the Explainable Artificial Intelligence (XAI) community on how to clearly differentiate these concepts [42] [206] [230] [410]. Given this ongoing debate and lack of standardization, in this work we adopt the term explainability to refer broadly to the overall process of making AI systems more understandable to humans.

Despite advances in these areas, GNNs often remain "black box" models. To address this issue, various post-hoc explainability methods have been introduced for graph neural networks [472], such as PGExplainer[374] and GNNExplainer [654], alongside data attribution techniques [588]. These approaches aim to provide insight into how GNNs make their predictions by identifying important input features or graph substructures. Nonetheless, while these methods offer some understanding, they still fall short of providing a comprehensive understanding of the model's internal workings [205, 4].

Our work addresses these limitations by proposing a novel model that integrates explainability directly into its architecture. Specifically, we build upon the Graph Transformer (GT) model [172]. This architecture offers the possibility of visualizing attention

maps, enabling us to illustrate how the model identifies relationships within graph structures. By examining these attention maps, we can verify whether the model focuses on graph regions consistent with known physics. To further enhance explainability, we replace the standard feed-forward layer of the Transformer with a Mixture of Experts (MoE) layer [528]. This design enables the model to distribute its outputs across specialized "experts," each targeting distinct data subsets, thereby providing a clearer understanding of how individual components contribute to the final decision. This enhanced transparency allows us to trace the model's reasoning pathways and verify their alignment with physical principles. We test our approach on Monte Carlo-simulated collision events, targeting hypothetical particles predicted by Supersymmetry (SUSY). These datasets include both rare SUSY signal events and dominant Standard Model (SM) backgrounds, following established ATLAS methodologies [29]. The classification task, distinguishing rare signal from high-rate background, mirrors the challenges typical in HEP. Our focus is not solely on achieving high performance but on validating the explainability of the model. By leveraging this controlled dataset, we demonstrate how our method identifies critical features and regions of the graph that align with the expected physics, fostering trust in the model's predictions. In this work, we emphasize that the explainability of machine-learning models is as crucial as their accuracy, particularly when exploring the potential for new physics. Our approach illustrates how explainable GNN-based methods can provide high predictive accuracy, reinforcing the connection between computational predictions and physical intuition. This focus on explainability ensures that our methodology is not only a powerful analytical tool but also a reliable framework for future discoveries in particle physics.

5.2.1 Related Work

Graph Neural Networks and Graph Transformers

GNNs are a specialized class of neural networks designed to process and analyze graph-structured data, with their key strength lying in their ability to learn representations that capture complex relationships among nodes in a graph. GNNs are extremely versatile and can be adapted to various graph-related tasks, including node and graph classification, regression, edge prediction, graph generation, and node clustering. They have found wide-ranging applications in particle physics [568], [536] and HEP, especially in particle tracking and reconstruction [170, 288]. The physics tasks at the LHC experiments have provided many potential applications where graph neural networks have been successfully applied [160]. Building upon the foundation of GNNs, GTs have recently emerged as a powerful graph learning method. GTs draw inspiration from the success of Transformers in natural language processing (NLP) [583] and computer vision (CV) [168]. GTs excel at handling dynamic and heterogeneous graphs, utilizing both node and edge features [172]. Various adaptations and expansions of GTs have demonstrated their superiority in addressing multiple challenges in graph learning, such as processing large-scale graphs [619]. In addition to these advancements, explainability methods for attention mechanisms have provided valuable insights into the inner workings of self-attention models, further enhancing their explainability [117].

Mixture of Experts

The Mixture of Experts (MoE, [528]) architecture offers a modular approach to scaling neural networks size. As models increase in size and complexity, they require substantial computational resources. MoE addresses this challenge by introducing expert specialization, where multiple subnetworks, or "experts", are trained to handle different regions of the input space. Instead of using a single, large model for all inputs, MoE incorporates a gating mechanism that dynamically assigns each input to a subset of experts, selecting the most relevant ones based on learned patterns [528]. This mechanism allows for task-specific specialization, allowing each expert to focus on different aspects of input data [173]. As a result, the architecture can scale more efficiently, allowing for higher model capacity without the need for proportionally increased computation for every input. By activating only a small number of experts at a time, MoE provides a computationally efficient solution for handling large and complex models [330]. Moreover, recent advancements in explainability for MoE architectures, such as analyzing expert utilization and routing behavior, have deepened our understanding of their decision-making processes and modality-specific specializations[418].

Explainability for High Energy Physics

Several explainability methods have been applied to HEP tasks, illustrating their utility across different domains. Models like XGBoost [101] and Deep Neural Networks (DNNs) have been interpreted using SHapley Additive exPlanations (SHAP) [373]. SHAP studies have proven effective to understand how signal events, characterised by the presence of a Higgs boson candidate, are distinguished from background noise through physically relevant kinematic variables [461]. Layerwise Relevance Propagation (LRP)[323] has been used to enhance the explainability of DNNs for classifying boosted Z boson jets [5]. Through LRP-based heatmaps, the method highlighted relevant detector regions. The application of LRP has also been extended to GNNs in the particle-flow reconstruction task [409]. By distributing relevance scores, the study quantified the importance of input features such as charge and energy, as well as the connections within graph structures. Additionally, data attribution techniques like TracIn[474] have been employed to identify impactful training data in the ATLAS datasets considered in our paper [29] This approach improved both model efficiency and explainability by refining training datasets and revealing influential patterns [588]. However, these methods often exhibit brittleness and lack reliability in challenging real-world scenarios. For instance, studies have shown that many post hoc explainability techniques fail fundamental sanity checks, raising concerns about their validity and robustness [4]. Other research has highlighted the fragility of interpretations, particularly when faced with slight changes in input or model parameters [205]. Attribution-based methods, in particular, have been criticized for being overly sensitive to model biases, making their insights less reliable in practical applications [688]. To address these challenges, we propose intrinsic techniques that integrate explainability directly into the model architecture and learning process, enabling the analysis of the model's internal decision-making mechanisms for complex HEP tasks without relying on post-hoc explanations.

5.2.2 Experimental setup

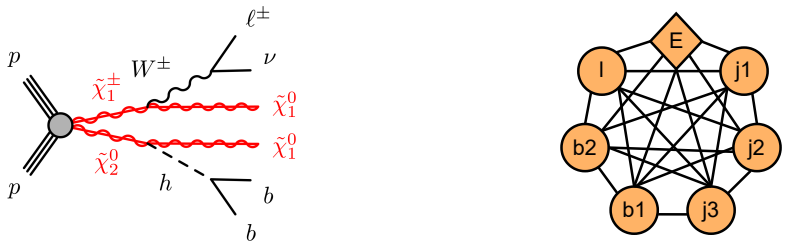
Dataset

The SUSY dataset, derived from the ATLAS publication [29] and available in the CERN open data repository (https://opendata.cern.ch/record/28100), contains Monte Carlo simulated collision events for a set of signal models (Signal) and SM backgrounds – where the two major sources are hereby considered (Background). Details on the two types of events are given below:

- **Signal**: Collision signal events are depicted in Fig. 5.20a. Two parent particles (named chargino, $\tilde{\chi}_1^{\pm}$, and next-to-lightest neutralino, $\tilde{\chi}_2^0$) are produced at the collision point and promptly decay into a W and Higgs boson, respectively, and the dark matter candidate (the lightest-neutralino, $\tilde{\chi}_1^0$). The W boson decay immediately in a lepton (l) and a neutrino (ν), and the Higgs boson in a pair of b-quarks.

- **Background**: SM background events represent the processes of pair production of two top quarks ($t\bar{t}$) and the production of a top quark and a W boson (Wt or single top). Other sources of SM backgrounds considered in the ATLAS paper are not considered here as potentially more distinguishable from signal and anyway contributing for less than 20% of the total background. Both top-quark processes mimic the signal as they result in similar final particles.

Signal models are parametrised as a function of two variables, the mass of the SUSY parent particles (equal, according to the model) and the mass of the dark matter particle. The mass value ranges between 125 GeV (the value of the Higgs mass) and 1000 GeV for the former, and 0 and 400 GeV for the latter, although the highest statistics models are in the region where the mass splitting between parent and dark matter particles is close to the Higgs mass, as directly targeted by the ATLAS paper. The kinematic of the signal events depends on the choices of masses, and each choice represents a different model. If SUSY were realised in nature, only one choice of mass parameters would

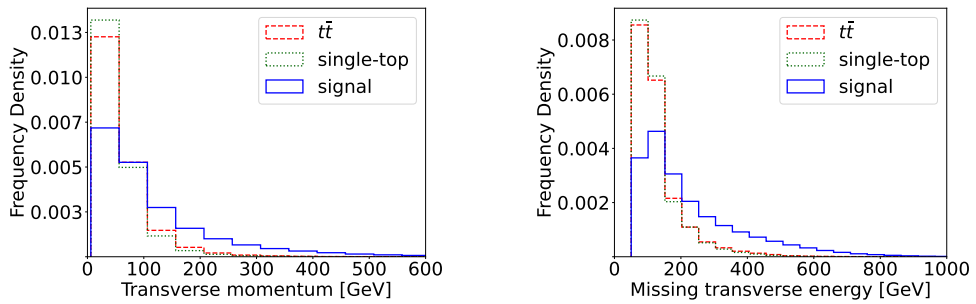


(a) **(b)**

Figure 5.20. (a) Diagram of the SUSY signal process, showing chargino and neutralino decaying into W and Higgs bosons, with leptons, neutrinos, and b-quarks in the final state. (b) Schematic representation of a particle collision event modeled as a fully connected graph. Nodes represent reconstructed particles and objects: jets (j1, j2, j3), b-tagged jets (b1, b2), lepton (l), and missing transverse energy (E). The diamond shape for E indicates that it is not a directly reconstructed particle but an inferred quantity.

be the correct one. Since this is unknown a priori, all models are mixed together with equivalent probability.

The main task involves detecting the rare signals amidst a large background of SM processes. In real data, signal events are expected to be well below 0.1% of the total background, even after dedicated selections are imposed. To distinguish these events, we rely on kinematic features that offer significant discriminative power. We represent each collision event as a fully connected graph G with 6-7 nodes (N), where each node represents a particle and contains up to 6 kinematic features. Mathematically, each event is encoded as a feature matrix $X \in \mathbb{R}^{N\times 6}$, where rows correspond to particles and columns to their features. The scheme for particle collisions and their corresponding features are depicted in the graph in Fig. 5.20b and, in detail, in Table 5.3. With reference to the Table, each row represents a distinct particle that was reconstructed in the detector. The first rows correspond to the jets (*j1, j2, j3*), where jets are produced by the hadronisation process of quarks. The *b1* and *b2* represent the jets originated from the b-quarks present in signal and background events, and are a subset of the generic jets. For events with 6 nodes, *j1* and *j2* are equivalent to *b1* and *b2*. Events with 7 nodes are those with an extra jet, not depicted in Fig. 5.20a. This categorization is made as the presence of a third jet is expected to offer additional discriminative power. The *lepton* row, l, refers to either a muon or an electron, depending on the particle identified. Finally, the *energy* row (E) represents the so-called "missing" energy, used as a proxy for particles invisible to the detector such as neutrinos and dark matter candidates. The dataset columns describe various physical properties of the particles. Feature 1, p_T, represents the transverse momentum magnitude for particles, while for the *energy* row, it is the E_T^{Miss} quantity (called missing transverse energy). Features 2 and 3 are the pseudorapidity (η, related to the angle of a particle relative to the beam axis θ in collision events) and the azimuthal angle (ϕ) of jet and lepton objects, respectively. Only the azimuthal angle of E_T^{Miss} can be defined due to the unknown longitudinal momentum in head-on collisions at the LHC. Feature 4 is referred to as "quantile" and quantifies the likelihood for a jet to be identified as originating from a b-quark or not according to the ATLAS b-tagging identification procedure. Feature 5, *b1m* and *b2m*, denote the mass of the b-jets calculated from jet component four-vectors. The final column is σ_{ETMiss}, which is implemented as a feature of the missing transverse energy quantity and directly



(a) **(b)**

Figure 5.21. Distributions of (a) p_T of lepton and (b) for the E_T^{Miss}, comparing signal and the two main background processes.

related to its resolution [29]. In Fig. 5.21 (a) and (b), key features are shown for signal and the two kind of background events considered here, normalised to the unit area. Top-pair and single-top backgrounds are added together as kinematically similar. All signal models are mixed together, with models defined by (low) high mass particles contributing mostly to the bulk (tails) of the distributions.

Table 5.3. Kinematic properties of particles.

Particle	F1	F2	F3	F4	F5	F6
j1	p_T^{j1}	η_{j1}	ϕ_{j1}	q_{j1}	-	-
j2	p_T^{j2}	η_{j2}	ϕ_{j2}	q_{j2}	-	-
j3	p_T^{j3}	η_{j3}	ϕ_{j3}	q_{j3}	-	-
b1	p_T^{b1}	η_{b1}	ϕ_{b1}	q_{b1}	m_{b1}	-
b2	p_T^{b2}	η_{b2}	ϕ_{b2}	q_{b2}	m_{b2}	-
lepton	p_T^{l}	η_l	ϕ_l	-	-	-
energy	E_T^{Miss}	-	ϕ_{ETMiss}	-	-	σ_{ETMiss}

Mixture of Experts Graph Transformer

Given a graph G, representing a collision event as explained in Section 5.2.2, our task is to predict a label y such that $y = f(G)$. Specifically, the label y encodes the class of the collision event that we aim to predict – *Signal* or *Background* – using a neural network model f. To ensure that the predictions are as interpretable as possible, we design the model f with intrinsic explainability in mind and model it as a GT architecture [172] enhanced with a MoE. The GT processes graph-structured data using self-attention and feed-forward networks, while the MoE enables parameter subsets to specialize in distinct data aspects. This architecture ensures intrinsic explainability by using attention to quantify node relationships. Figure 5.23 illustrates the model, highlighting attention maps and the MoE's role in capturing dependencies and specialization.

We test our proposed model on the Monte Carlo simulated collisions events for signal and SM background events described above. In this setup, each graph event G is characterized by a feature matrix $X \in \mathbb{R}^{N\times 6}$, where N is the number of nodes in the graph and each row vector x_i represents a node embedding with its features. The GT updates nodes representations using a sequence of layers consisting in a self-attention mechanism followed by a feed-forward block. After the initial addition of Laplacian positional encodings [44][356], [544], for each node x_i, the attention mechanism computes a weighted sum of value vectors V, where the weights are determined by scaled dot-product attention [172]. Let Q, K, V denote the query, key, and value matrices derived from the input X via learned linear projections W_Q, W_K, W_V. The attention for x_i is:

$$\text{Attention}(x_i) = \text{softmax}\left(\frac{q_i K^\top}{\sqrt{d_k}}\right)V \tag{5.6}$$

where $q_i = x_i W_Q$ is the query vector for x_i, $K = XW_K$ is the matrix of key vectors, $V = XW_V$ is the matrix of value vectors, and d_k is the dimensionality of the key vectors used for scaling. We refer to $\text{softmax}\left(\frac{q_i K^\top}{\sqrt{d_k}}\right)$ as "attention weights", as they serve to

encode the relationships between nodes, effectively quantifying their interactions and dependencies. This view offers an interpretable framework for understanding how the model processes information and makes decisions, which we describe more in detail in Section 5.2.4. The GT employs multi-head attention, extending the attention mechanism by employing H independent attention heads, each with its own set of projections W_Q^h, W_K^h, W_V^h for $h = 1, \ldots, H$. The outputs of these heads are concatenated and linearly projected back to the original dimensionality. For x_i, the multi-head attention is:

$$\text{MultiHead}(x_i) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)W_O, \qquad \text{with} \qquad \text{head}_h = \text{softmax}\left(\frac{q_i^h (K^h)^\top}{\sqrt{d_k}}\right) V^h \tag{5.7}$$

where $q_i^h = x_i W_Q^h$, $K^h = X W_K^h$, and $V^h = X W_V^h$. The output projection W_O combines the H heads into a single representation. After updating the node representations according to Eq. (5.7), the node embeddings are passed through a feed-forward network (FFN), consisting in a sequence (usually two) learnable linear projections with an intermediate nonlinearity:

$$x_i \mapsto W_2(\text{ReLU}(W_1(x_i) + \beta_1)) + \beta_2 \tag{5.8}$$

where x_i represents a generic node embedding, W_1, β_1, W_2, b_2 represent learnable weights and biases, and $ReLU$ is a nonlinear activation function.

In order to enhance explainability, we also replace the original FFN with a MoE block, as described next. The MoE architecture consists of n expert networks, denoted as $E_1, \ldots, E_n$, and a routing network r. Each expert is a neural network defined as in Eq. (5.8) with its own parameters. The routing network r determines a set of weights for the expert outputs based on the updated node representation, thus assigning each node of the graph to one or more expert networks. The output y of the MoE block is computed as a weighted combination of the expert outputs:

$$y = \sum_{i=1}^{n} r(x)_i E_i(x).$$

The MoE enhances explainability by allowing us to identify which experts contribute most to the prediction for a given node, providing insights into the model's decision-making process. Additionally, the MoE optimizes computational efficiency by leveraging the sparsity of the gating network's output. When $r(x)_i = 0$, we bypass the computation of $E_i(x)$, thus saving computational resources. We follow (Shazeer et al, 2017)[528] for the design of the routing network, and implement noisy *top-K* gating with a load balancing loss to prevent imbalanced routing choices that would lead to the training of a small subset of experts, a well-known issue for MoE models. Details about the routing are given in S1 of Supplementary Materials. The overall architecture is further depicted in Fig. 5.22 and includes the GT stack and the classification head tailored to the task.

5.2.3 Results

We conducted experiments to evaluate the performance of four architectures—Graph Convolutional Network (GCN), MLP, GT, and our model described in Section 5.2.2, the Mixture of Experts Graph Transformer (MGT)—on the SUSY dataset. The GCN architecture, uses two convolutional layers with hidden channels of size 80 and 40,

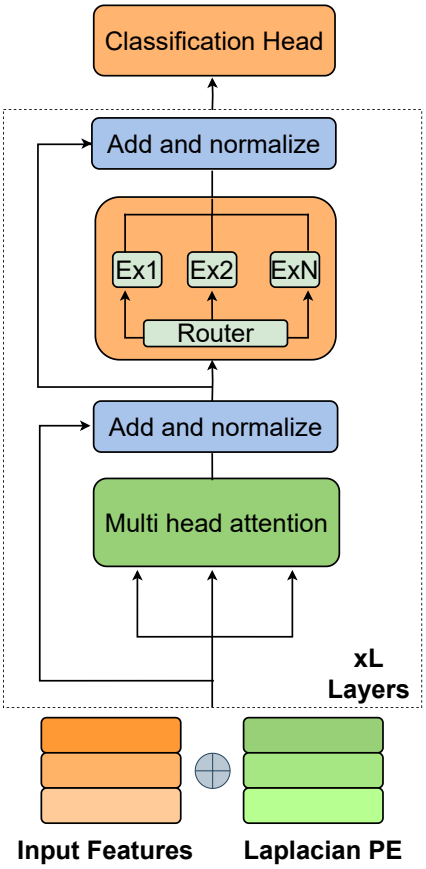


Figure 5.22. Detailed illustration of the proposed architecture incorporating multi-head attention with an MoE block. The model begins with Laplacian positional encoding for the input features, followed by multi-head attention and normalization. This structure is repeated across L layers. The MoE block, driven by a gating network, assigns inputs dynamically to specialized expert networks. A classification head processes the final representation to produce predictions.

respectively. It employs global mean pooling for aggregation and concludes with a linear layer to output the class predictions. The GCN results show promising differentiation between signal and background, aligning with key kinematic features of the dataset, with detailed analysis, including feature importance and event correlations, provided in S4 of Supplementary Materials. The MLP model also consists of two hidden layers, with dimensions 80 and 40, respectively, and takes the input graph as a flattened vector. The GT architecture is composed of two layers, each with 2 attention heads, Laplacian positional embeddings of size 4 and a hidden size of 80. The MGT model builds upon the GT architecture with a hidden size of 80, 2 attention heads, 6 expert networks each with an expert size of 20, and 2 layers. It also employs a gating mechanism with a weight load factor of 1 for expert selection.

The experiments were conducted using ten runs, with a different random seed for each run, to ensure robust evaluation of model performance. The results, quantified in terms of accuracy, precision, recall, F1-score, and AUC are reported in Table 5.4. The MoE architecture outperforms the other models, achieving superior accuracy and

overall performance metrics. This highlights the effectiveness of MoE in leveraging expert networks for enhanced predictive capabilities. Further analysis on training is provided in S3 of the Supplementary Materials.

Table 5.4. Performance comparison of different architectures (MLP, GCN, GT, and MGT) on the SUSY dataset. Results are reported as the mean ± standard deviation over ten runs with different random seeds. Metrics include Accuracy, Precision, Recall, F1 score, and AUC.

Model	Accuracy	Precision	Recall	F1	AUC
GCN	0.750 ± 0.0022	0.779 ± 0.2549	0.700 ± 0.0427	0.736 ± 0.0131	0.832 ± 0.0134
MLP	0.829 ± 0.0015	0.826 ± 0.0054	0.835 ± 0.0101	0.830 ± 0.0027	0.913 ± 0.0017
GT	0.849 ± 0.0059	0.850 ± 0.0062	0.849 ± 0.0059	0.849 ± 0.0060	0.928 ± 0.0057
MGT	$\boldsymbol{m}0.852 \pm \boldsymbol{m}0.0005$	$\boldsymbol{m}0.851 \pm \boldsymbol{m}0.0008$	$\boldsymbol{m}0.849 \pm \boldsymbol{m}0.0016$	$\boldsymbol{m}0.854 \pm \boldsymbol{m}0.0004$	$\boldsymbol{m}0.929 \pm \boldsymbol{m}0.0039$

5.2.4 Discussion

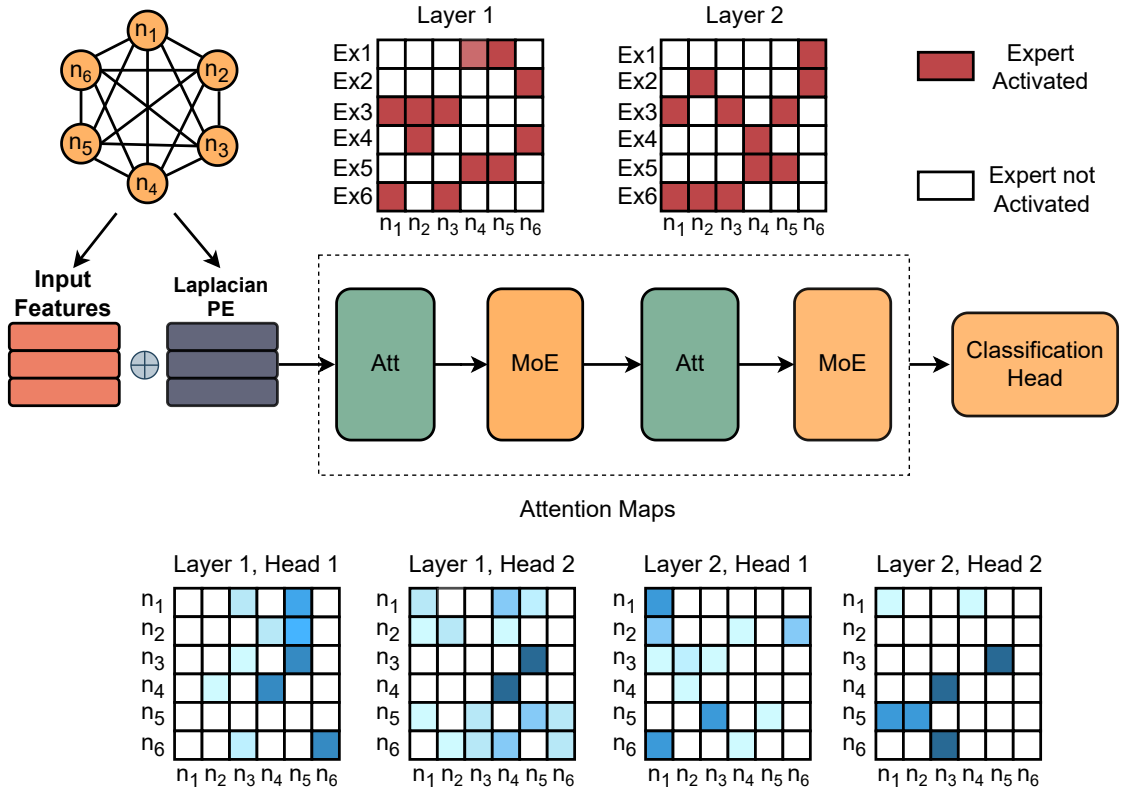


Figure 5.23. Overview of the proposed Transformer-based model architecture: the image illustrates an example of the model, which takes as input a graph. The graph is processed by the model, comprising various blocks: **Att**, which is the Multi-Head Attention block, and **MoE** blocks. The visualization includes attention maps derived from the Multi-Head Attention mechanism and the activation patterns of the experts for a single collision event example.

The attention mechanisms and expert specialization in the MoE model provide valuable insights into its explainability and performance. This section discusses the

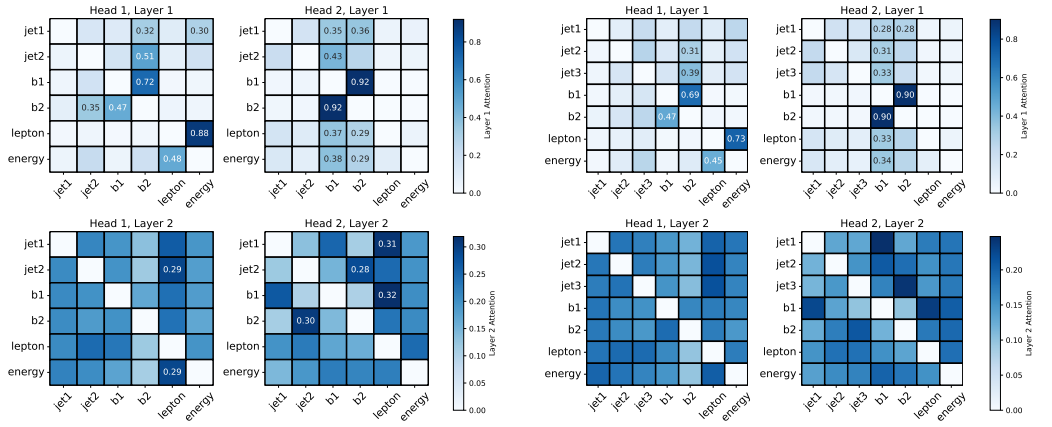
detailed behavior of attention scores and the unique specialization patterns exhibited by expert networks. The overall architecture of the proposed Transformer-based model is illustrated in Figure 5.23, providing a detailed view of how the graph input is processed through the MoE blocks.

Analysis of the Attention Heads

Attention scores provide significant insights into the model's internal mechanisms by refining node and feature representations while capturing relationships and dependencies within the graph. In this process, attention acts as a guiding mechanism, directing how information is aggregated and propagated through the network. In an attention map, the attention weights are organized such that each row corresponds to a query (i.e., the element doing the attending), and each column corresponds to a key (i.e., the element being attended to). To interpret the model, we conduct a detailed analysis of the attention scores by examining multiple subsets of the dataset, including the training set, test set, as well as correctly classified signal events (true positives) and correctly classified background events (true negatives). For each of these groups, we compute the attention matrix for each head, averaging the values across the data to uncover global trends in attention behavior. The images shown in the following (Fig. 5.24, Fig. 5.25, Fig. 5.26) are divided into two columns, with 6-node graphs on the left and 7-node graphs on the right. Each row represents the attention given by a specific node to all other nodes in the graph. Attention scores above the 85th percentile, computed across all layers and heads, are visualised to highlight the most significant interactions while reducing noise. By visualizing the attention maps, we can diagnose how attention guides the formation of feature representations, shedding light on the underlying mechanisms of the model. Fig. 5.24 shows the resulting image for the test set. Attention in the first layer displays higher magnitudes and broader distributions, capturing diverse relationships within the graph. In contrast, the second layer shows reduced magnitudes, indicating a refinement of features. This hierarchical approach reflects the model's process of first aggregating general dependencies and then fine-tuning these into task-relevant representations. The attention focus in head 2 is dominated by core features of *b1*, *b2*, whereas in head 1 the *energy*, *lepton* plays a more central role, with *b1*, *b2* remaining of secondary relevance. This behaviour is consistently observed across both graph types. This is in excellent agreement with the physics expectations: the origin of the *b1*,*b2* is a Higgs boson in signal that is not present in background, and as such is clearly distinctive; the missing transverse energy, in particular when related to the lepton object, is expected to be the next distinctive feature because it originates from the dark matter particles and the neutrino in signal, and only from the neutrino in background, with lepton and neutrino originating from the decay of the W boson (both in signal and background).

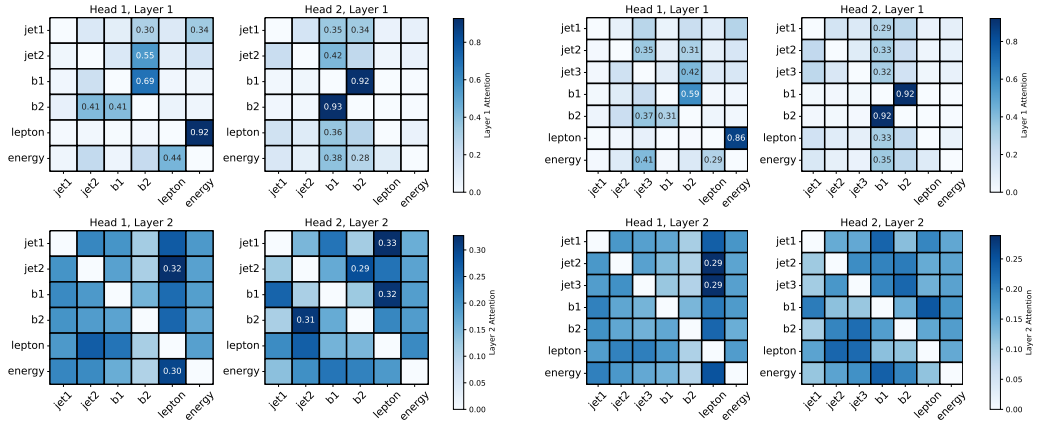
The above considerations are even better depicted when considering correctly classified signals (Fig. 5.25), where attention strongly concentrates on *b1*, *b2*, and *energy* nodes, highlighting their importance in distinguishing signal events. In contrast, for correctly classified backgrounds (Fig. 5.26), attention is more evenly distributed across nodes, reflecting the broader and less distinctive feature patterns typical of background events. Finally, we note how the additional jet node in 7-node graphs receives some attention in the second layer for signal events, indicating a refinement of features based on its correlation with the other jets in the event. More specifically, the attention score

of *j3* in 7-node graphs can be explained by the fact that the feature η is expected to have some distinctive power, being in average lower for background and larger for signal. The quantile of such extra jet is also an additional distinctive feature.



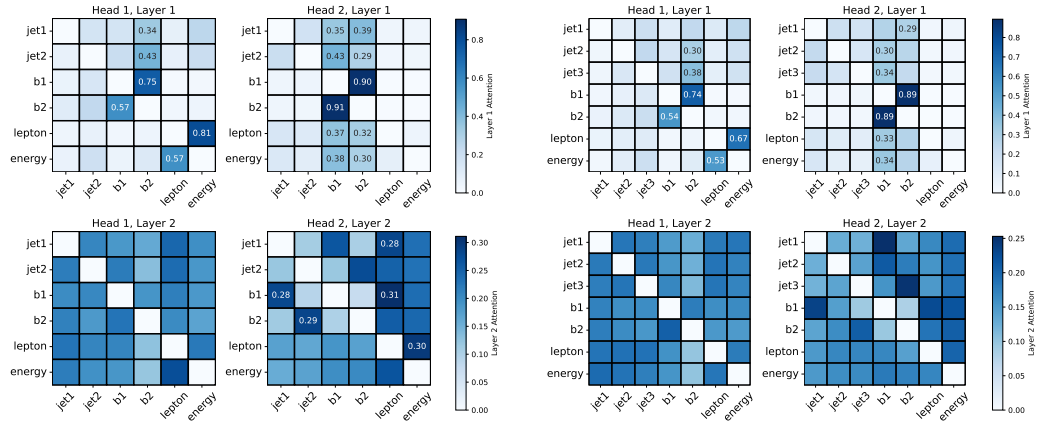
(a) 6 nodes. **(b)** 7 nodes.

Figure 5.24. Attention maps for test set.Each image shows the averaged attention scores for a specific attention head. The color scale ranges from white (low attention) to dark blue (high attention), visualizing only attention scores above the 85th percentile (computed across all layers and heads) to highlight the most significant interactions. The left column displays maps for 6-node graphs; the right column for 7-node graphs.



(a) 6 nodes. **(b)** 7 nodes.

Figure 5.25. Attention maps for correclty classified signals (true positives). Each image shows the averaged attention scores for a specific attention head. The color scale ranges from white (low attention) to dark blue (high attention), visualizing only attention scores above the 85th percentile (computed across all layers and heads) to highlight the most significant interactions. The left column displays maps for 6-node graphs; the right column for 7-node graphs.



(a) 6 nodes. **(b)** 7 nodes.

Figure 5.26. Attention maps for correctly classified backgrounds (true negatives). Each image shows the averaged attention scores for a specific attention head. The color scale ranges from white (low attention) to dark blue (high attention), visualizing only attention scores above the 85th percentile (computed across all layers and heads) to highlight the most significant interactions. The left column displays maps for 6-node graphs; the right column for 7-node graphs.

Analysis of Experts Specialization

Unlike traditional neural network approaches that rely on a single monolithic network, MoE introduces a dynamic mechanism which can be helpful in processing and interpreting complex data. The key motivation behind employing MoE in our GT model is to understand how different expert networks specialize in processing distinct aspects of the interaction graph. We test node specialization which specifies how the model processes inputs by dividing the workload among its components. Each specialized expert becomes highly tuned to a certain type of input (in our case the type of node), allowing it to extract and represent the most relevant information from that subset. Using the same dataset of signal and background events, the specialization patterns of the MoE block for layers 1 and 2 is studied. Fig. 5.27) highlights the distinct ways the model processes the features of different nodes. The MoE block of layer 1 appears to focus on extracting fundamental distinctions between node types, with strong specialization and minimal overlap between experts. The MoE block of layer 2, on the other hand, shows slightly more overlap among experts, suggesting that it operates at a higher level of abstraction, combining features and resolving correlations between particle types to refine the classification. In Tables 5.5 and 5.6, we present a summary of the specializations of the various experts, highlighting the nodes where the experts exhibit the highest activation. In more detail, for the MoE block of layer 1, Experts 2 and 4 focus on *lepton* and *energy* nodes, which are the quantities closely tied to the W boson. As highlighted before, in background events, the missing energy arises solely from the neutrino produced in the decay of the W boson whilst in signal events missing energy also originates from dark matter candidates. The specialization of Experts 2 and 4 indicates that the model identifies the combined behavior of *lepton* and missing energy features as a key discriminator. Together, these quantities encode differences

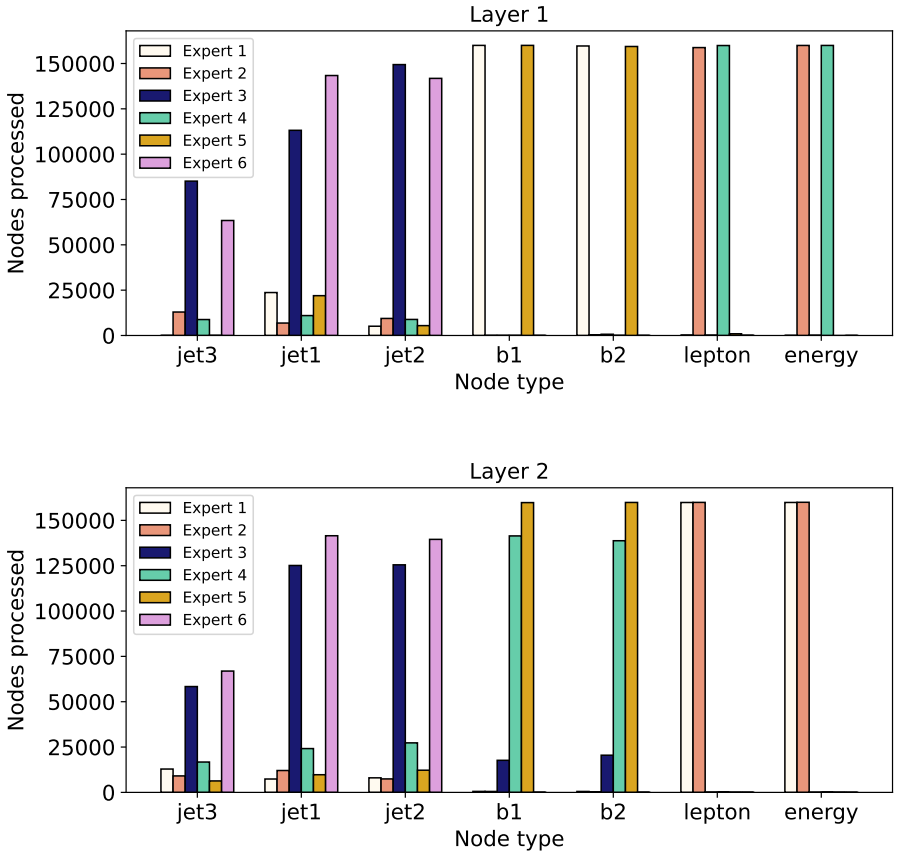


Figure 5.27. Visualization of node specialization in the MoE architecture for layers 1 and 2. The bar charts show the number of nodes processed by each expert for different node types (*jet1*, *jet2*, *jet3*, *b1*, *b2*, *lepton*, *energy*).

between signal and background events, such as the overall energy imbalance or the transverse momentum carried away by undetected particles. The paired specialization of Experts 1 and 5 on the two b-jets suggests that the model learns to process b-jets as a correlated entity rather than treating them in isolation. This approach is consistent with the physics. Experts 3 and 6 demonstrate specialization in *jet1*, *jet2*, and to a lesser degree, *jet3*. This pattern can be explained by the structure of the dataset. For 6-node graphs, *jet1*, *jet2* are equivalent to *b1*, *b2*, hence the same specialization of Experts 1 and 5 is expected. For 7-node graphs, *jet3* acquires an added distinctive value. The fact that it is not consistently present across all events limits the opportunities for these experts to learn from its features. The MoE block of layer 2, on the other hand, shows slightly more overlap among experts. This may suggest that layer 2 operates at a higher level of abstraction—combining features and resolving correlations between particle types to refine the classification—the qualitative nature of this observation does not allow us to make a definitive quantitative claim.

Table 5.5. Expert Specialization for layer 1.

Expert Number	Specialization	Physical Explanation
1, 5	*b1*, *b2*	Experts focus on the b-jets as a correlated entity. In signal events, these arise from Higgs boson decay, but in background events, they do not.
2, 4	*lepton*, *energy*	Experts specialize in detecting lepton and missing energy features, key indicators for distinguishing signal from background. In signal, missing energy also originates from dark matter particles, unlike the background.
3, 6	*jet1*, *jet2*, *jet3*	Equivalent to *b1* and *b2* in 6-node graphs, but in 7-node graphs, the additional jet (*jet3*) introduces distinct kinematic features that alter the specialization

Table 5.6. Expert Specialization for layer 2.

Expert Number	Specialization	Physical Explanation
1, 2	*lepton*, *energy*	Similar to Experts 2 and 4 in Layer 1
3, 6	*jet1*, *jet2*, *jet3*	Similar to Experts 3 and 6 in Layer 1, but with less specialization
4, 5	*b1*, *b2*	Similar to Experts 1 and 5 in Layer 1, but with less specialization

5.2.5 Conclusion

In this work, we introduced a novel Mixture of Experts layer for graph data, aiming to enhance both explainability and performance in high-energy particle physics analysis. The proposed model, evaluated on a HEP dataset of simulated events from a search for dark matter in SUSY models published by the ATLAS experiment, achieved superior predictive accuracy compared to baseline models. This improvement underscores the effectiveness of leveraging expert subnetworks to address the task-specific complexities inherent in high-energy physics data. Beyond its strong predictive performance, the model made significant strides in addressing the critical need for explainability in high-energy physics. By visualizing attention maps and analyzing expert specializations, we identified key features and their influence on classification outcomes, ensuring that the model's decision-making aligns with established physical principles. This ability to trace predictions back to interpretable components not only enhances trust in the model but also provides a deeper understanding of the physics involved. Despite these achievements, the study has some limitations that warrant further exploration. The scalability of the MGT model to larger datasets or graphs with more nodes remains an open question, as does its ability to generalize across a wider range of high-energy physics tasks. Moreover, future validations of these results on other datasets and larger architectures will be critical in establishing their robustness and broader applicability. Future research could extend this framework to other types of datasets and particle physics phenomena, testing its adaptability and robustness across diverse scenarios. Furthermore, these results are particularly promising in relation to the application of advanced mechanistic explainability techniques, such as Sparse Autoencoders [122], to

this type of data. The integration of emerging techniques for automated explainability, including the use of Large Language Models [56], could further enhance the accessibility and dynamism of the model's interpretations, amplifying its utility for scientific discovery. By combining high predictive performance with enhanced explainability, this work lays the groundwork for advancing explainable machine-learning methodologies in particle physics. Building on these foundations promises to unlock further potential in addressing the challenges of analyzing complex scientific data.

Data Availability Statement

The data used is public and released under Creative Commons Zero v1.0 Universal license. It is fully accessible via opendata https://opendata.cern.ch/record/28100, where a description of the content is included. The generation and simulation setup are fully detailed in the cited ATLAS paper (10.1007/JHEP12(2023)167). The code for the model is available at https://github.com/DonatellaGenovese/Mixture-of-Expert-Graph-Transformer.

Supplementary materials

S1 Mixture of Experts Routing and Balancing loss

The Mixture of Experts (MoE) architecture provides a more efficient way to allocate resources by specializing subnetworks (Experts), enabling the possibility for sparse computing. This design is a core component of the Mixture of Experts Graph Transformer (MGT) introduced in Section 5.2.2, where it replaces the standard feed-forward networks in the Graph Transformer (GT) layers. By integrating MoE, the MGT achieves both computational efficiency and enhanced explainability.

At the core of the MoE layer in the MGT is the concept of specialization, where multiple expert networks learn to focus on different regions of the input space. Instead of employing a monolithic model to process every input, MoE uses a gating mechanism that dynamically selects the most relevant experts for each node representation. This dynamic selection not only reduces the computational cost by utilizing a sparse subset of experts for each input but also enables detailed interpretability by identifying which experts were most influential in the decision-making process.

In the MGT architecture (Section 5.2.2), the MoE layer is applied to updated node embeddings after the attention mechanism. The routing network, as detailed below, determines the top-k experts for each node and provides the weights that combine their outputs. The gating mechanism employs Noisy Top-K Gating [528], and its implementation aligns closely with the MGT design goals outlined in the main text. The gating function used in the MoE layer is:

$$G(\hat{x}^l) = \text{Softmax}(\text{KeepTopK}(H(\hat{x}^l), k))$$

where

$$H(\hat{x}^l)_i = (\hat{x}^l \cdot W_g)_i + \text{Standardnormal}() \cdot \text{softplus}(\hat{x}^l \cdot W_{noise})_i$$

Here, W_g is the router matrix,$StandardNormal()$ is a random vector of dimensions $|E|$ with values drawn from a standard normal distribution, W_{noise} is a trainable weight

matrix and the function KeepTopK$(v,k)_i$ is defined as:

$$\text{KeepTopK}(v,k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the } k \text{ highest value components of } v \\ -\infty & \text{otherwise} \end{cases}$$

Each expert is defined as follows:

$$E(x) = \text{Dropout}(W_2(\text{LeakyReLU}(W_1(x) + \beta_1)) + \beta_2) \tag{5.9}$$

The input node is routed to the experts corresponding to the k non zero components of its gating function output.
The result of the expert's computation is combined as follows:

$$y = \sum_{i=1}^{k} G(x)_i E_i(x) \tag{5.10}$$

To ensure the even utilization of the experts, an additional loss component for load balancing is needed.
It is computed as follows:

$$L_{\text{load}}(X) = w_{\text{load}} \cdot \text{CV}(\text{Load}(X))^2 \tag{5.11}$$

With w_{load} being an hyperparameter factor that will control the weight of the load balancing loss with respect to the criterion, CV being the coefficient of variation and X being a batch of inputs.
The loads of the experts are computed as follows:

$$\text{Load}(X)_i = \sum_{x \in X} P(x,i) \tag{5.12}$$

where

$$P(x,i) = \Phi\left(\frac{\left(x \cdot W_g\right)_i - kth\_\text{excluding}\,(H(x),k,i)}{\text{Softplus}\left((x \cdot W_{\text{noise}})_i\right)}\right) \tag{5.13}$$

Here, Φ is the cumulative distribution function of the standard normal distribution, the term $kthexcluding(H(x),k,i)$ stands for the kth highest component of $H(x)$, excluding component i.

S2 Ablation Study for Mixture of Expert Graph Transformer

An ablation study is performed in order to understand the function of model's components and the impact that their configurations has on performance. The performance is assessed by using accuracy, precision, recall and f1 score metrics, in addition, the specialization behavior of the MoE layer is studied. The dataset used in this section will be a portion of the complete one, containing 20000 signal events, 10000 background ttbar events and 10000 background singletop events. The components studied are the following:

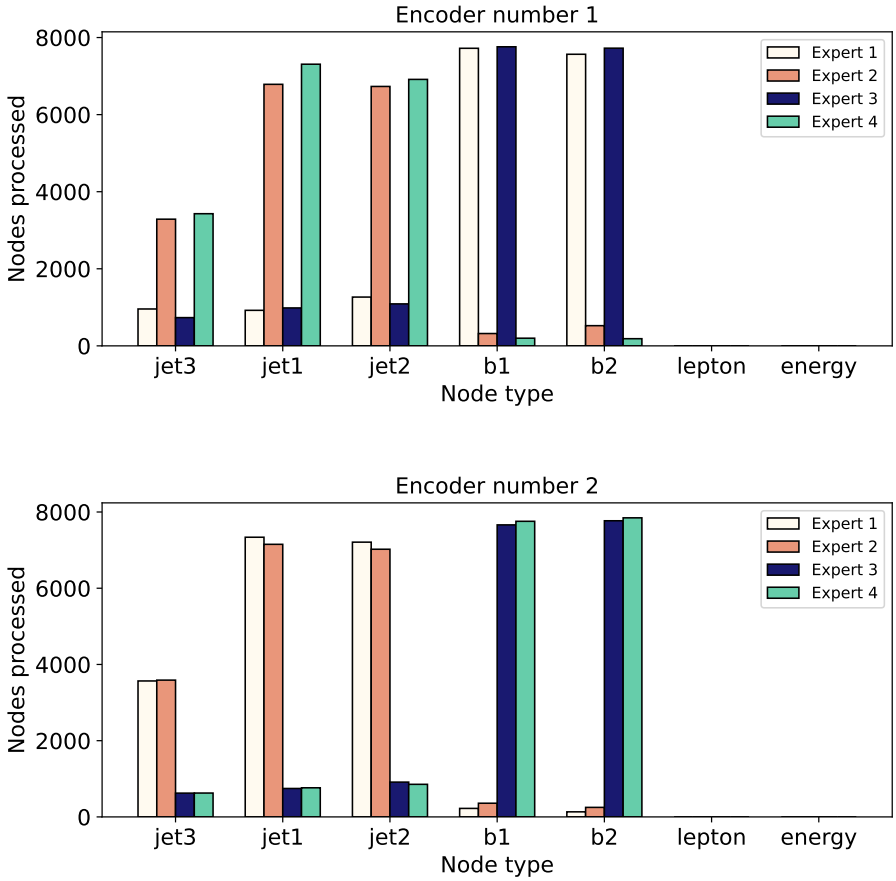


Figure 5.28. Expert specialization pattern with 4 total experts, top-2 routing, 2 layers

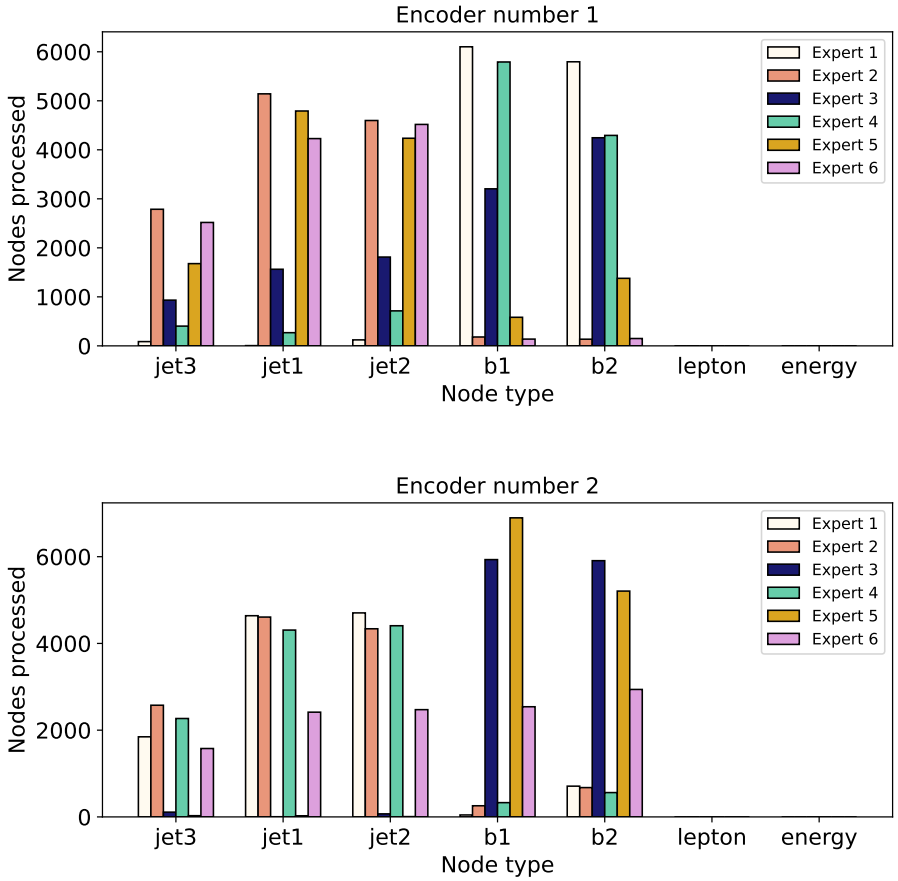


Figure 5.29. Expert specialization pattern with 6 total experts , top-2 routing, 2 layers

- **Weight of the load balancing loss**: The factor by which the load balancing loss is multiplied, before it being added to the chosen training criterion
- **Learning rate**: Learning rate configuration, including the parameters of the scheduler function
- **Mixture of Experts layer configuration (number of experts, k-routing, expert size)**: Respectively: how many total experts the MoE layer has, how many experts will be active for each token, the size of the matrices that compose an expert
- **Attention layer configuration (hidden size, number of heads)**:Respectively: size of the graph's embedding and attention matrices, number of heads for the multi-head attention layer
- **Dropout probability**: Probability of dropout after the attention layer and inside the experts subnetwork
- **Number of layers**: How many encoder layers the model will be composed of.

Changing the weight of the load balancing loss does impact the experts utilization behaviour significantly. When setting a value of zero, the experts are not evenly utilized and their ability to specialize is compromised. This hyperparameter's ideal value is 1, which allows the model to even out experts utilization quickly and ensure that the experts can specialize, values higher than one do not grant any tangible improvements.

Adjusting the learning rate has a notable impact on performance results and experts specialization pattern, the higher the learning rate is, the more sharp and less noisy the specialization pattern becomes. On the other hand, having a learning rate too high introduces instability in the training process that results in performance loss.

The number of total experts, experts size and routing choice have no measurable impact on the performance of the model, however, they heavily contribute to the resulting specialization pattern. The experts tend to specialize on particle types, more specifically on groups of highly correlated particles, on this specific dataset, the model configuration that achieves the sharpest specialization on the experts is using top-2 routing over 6 total experts. Let's consider k, the number of experts simultaneously active, n the number of total experts and l groups of highly correlated particles in the graph data, the specialization pattern will tend to a configuration in which, for every particle type node, there is a $k-$plet of active experts, the same $k-$plet will be active on nodes belonging to the same group of correlated particles. The closest n/k is to l, the sharper the specialization pattern becomes.
This behaviour is further tested with an additional ablation study done on the dataset. The graphs were truncated on the second to last node, leaving only jets and b-jets, 5 nodes in total for each graph in the dataset. In this case, the best performing model, with regard to specialization, has 4 total experts, two of them specialize on the three jets and the other two on b-jets Fig. 5.28; whereas, the previously tested 6 experts configuration does not manage to specialize correctly. Fig. 5.29

Changing the hidden size of the model does not result in significant differences, increasing the value of this hyperparameter too much leads to performance loss and run-to-run inconsistency. The number of heads also has no impact on performance or specialization

behaviour.

Adding layers beyond the first did not result in noticeable performance increase. Regarding the expert specialization behaviour, when testing the model with multiple layers, the MoE layers in deeper encoders saw a sharper and less noisy specialization pattern. This trend is seen in all of the MoE layer configurations tested.

Lowering dropout probability slightly improved the performance of the model, reducing, however, the sharpness of the experts specialization patterns. Also, the tendency of the specialization to be sharper at deeper layers reverses when setting the dropout probability to zero. When testing the model on the full dataset, the specialization capaibility did not suffer when lowering dropout probability, resulting only in a small accuracy boost without drawbacks.

Given the results of the ablation study it was decided to configure the model as follows:

- **Hidden size**: 80
- **Number of heads**: 2
- **Number of experts and routing**: 6, top-2 routing
- **Dropout probability**: 0
- **Number of layers**: 2

S3 Training details

Model	Architecture	Layers	Hidden Size
GCN	Graph Convolutional Network (GCN)	2 Convolutional Layers	Hidden channels: 80 and 40
MLP	Multilayer Perceptron	2 Hidden Layers	Hidden layers: 80 and 40
GT	Graph Transformer	2 Transformer Layers	Attention heads: 2; Hidden size: 80
MGT	Mixture of Experts Graph Transformer	2 Transformer Layers with Mixture of Experts Layer	Hidden size: 80; Attention heads: 2; 6 Experts with size 20 each

Table 5.7. Training configurations for the models used in this study.

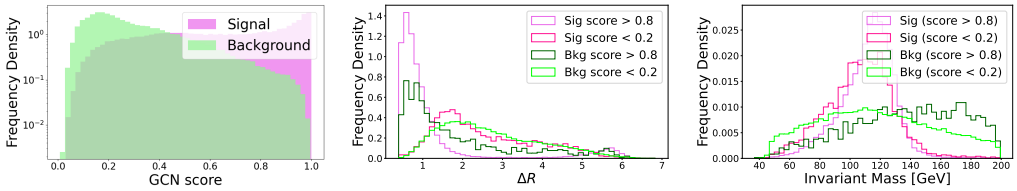
The training configurations for the four models used in this study are detailed in Table 5.7.

The parameters for the Mixture-of-Experts Graph Transformer (MGT) are selected based on the results of the ablation study presented in Section S2, which determined the optimal configuration for this architecture. For the other models, their parameters are chosen to align as closely as possible with those of the MGT to ensure comparability while respecting their architectural differences.

The dataset for training consisted of 800,000 events, with equal representation of signal and background events (400,000 each). Background events included 200,000 examples of pair production of top quarks ($t\bar{t}$) and 200,000 examples of single top and W boson production.

The dataset was first split into 80% training and 20% test sets. The training subset was further divided into 80% training and 20% validation sets. All models were trained for up to 60 epochs and using a batch size of 500. Validation loss was monitored during training with early stopping. Cross-entropy loss was employed as the primary loss function, with an additional load balancing loss included for the MGT to optimize expert utilization.

S4 Analysis of the GCN results



(a) **(b)** **(c)**

Figure 5.30. (a) Output score distribution for the GCN. Distributions of complex variables built from correlation of features: (b) ΔR *b1-b2* and, (c) invariant mass *b1-b2*.

GCN results show good differentiation between signal and background, as shown in Fig. 5.30(a). The two SM sources are mixed and treated as equivalent, since it was found that there is no dependence of the results on the proportion of single and pair top-quark background. To interpret the learning process in a simplistic way, we tested the impact of systematically hiding certain input features from the datasets. As expected according to the kinematic features of the signal and background events, the main drop in performance was obtained when removing from the training either the p_T features for *b1*, *b2*, or the features p_T of the *lepton* and the E_T^{Miss}. To also verify if the network effectively learns expected correlations within the graph, two key variables have been analysed depending on the output score: the spatial distance between *b1* and *b2* (ΔR), and the invariant mass of the two particles. From a physics perspective, in signal events *b1* and *b2* are originated in the decay of the Higgs boson, a massive particle with mass of 125 GeV. Hence, it is expected that the two decay products *b1* and *b2* are close to each other spatially, and that their invariant mass is around that of the Higgs boson. In the case of SM background, such correlations are not expected. Fig. 5.30 (b) and (c) show the distributions for ΔR and invariant mass for the *b1-b2* system for signal and background in case of high (> 0.8) and low (< 0.2) output score: the GCN effectively separates better signal models with close-by *b1*,*b2*, whilst background events with low score have larger spatial distance. In terms of the invariant mass, interpretability of the results is more difficult: however, we note how high-score background events (signal-like) tend to have higher invariant mass, which is less likely for signal event.

Finally, to further verify how relevant are the correlations on the network learning process, an additional test was performed by splitting the signal datasets into three

different groups, defined by the mass difference (Δm) of the parent and the dark matter particles, and performing a training run for each group separately. The three groups of samples were characterised by a $\Delta m = 150$ GeV (group 1), 200 GeV (group 2) and $> 500 GeV$ (group 3), respectively. The higher the mass difference, the higher the boost of the final decay products, which in turn is expected to result in a better distinction of the SUSY signal with respect to the background events. This was confirmed by the tests: with GCN, the AUC was found to be 0.71 ± 0.02 for group 1, 0.81 ± 0.01 for group 2 and 0.98 ± 0.01 for group 3. Using the MoE approach, AUC increases to 0.76 ± 0.02 for group 1 and to 0.84 ± 0.01 for group 2, remaining at 0.98 ± 0.01 for group 3. This demonstrates that (1) the kinematic of the signal events is exploited as expected and (2) the MoE model provides better distinction between signal and background also in the most challenging physics case.

5.3 Analysing the Residual Stream of LLMs under Knowledge Conflicts

Large language models (LLMs) have shown remarkable capability to memorise factual knowledge and solve knowledge-intensive tasks [460, 75, 575, 282, 16]. Nevertheless, the knowledge stored in their parameters (*parametric knowledge*) can be inaccurate or outdated. To alleviate this issue, retrieval and tool-augmented approaches have been widely adopted to provide LLMs with external knowledge (*contextual knowledge*) [295, 333, 616, 523]. However, such contextual knowledge can include information that conflicts with the parametric knowledge of the model, which may result in undesired behaviour; for example, the model can rely on inaccurate information sources and produce inaccurate generations [379, 626, 554, 599, 255, 677].

Prior research found that LLMs tend to prefer contextual knowledge (e.g. retrieved passages) over their parametric knowledge [554, 626]. However, in more general applications, LLMs should retain the ability to use parametric knowledge when presented with incorrect or undesirable information [94, 93, 701, 379, 684]. To achieve this goal, LLMs are expected to acknowledge the existence of conflicts, allowing them to alert the user while keeping the decision-making process under the user's control for further action. Existing works investigate the fine-tuning and prompting-based strategies to detect knowledge conflicts [599]. These methods need additional interactions with the model, e.g., by asking the LLMs to examine the conflicts sentence by sentence [599], which may result in high latency times and prevent practical applications of these models. Additionally, they do not provide insight into how LLMs internally detect and resolve conflicts.

In this work, we analyse the residual stream [175, 435] in LLMs to better understand their behaviour when knowledge conflicts arise, especially between parametric knowledge and contextual knowledge. Our probing experiments on the residual stream indicate that the signal of knowledge conflict rises from the intermediate layers (e.g., the 13th layer of Llama3-8B). Utilising this signal, a simple logistic regression model can achieve 90% accuracy in knowledge conflict detection without modifying the input and parameters of LLMs while introducing only a negligible computation overhead. Moreover, we also observe that the residual stream exhibits different patterns starting from the middle layers (e.g., the 17th layers of Llama3-8B) when the model takes different source information to resolve the conflict. For example, when the model uses contextual knowledge, the residual stream exhibits a significantly more skewed distribution compared with when it uses its parametric knowledge.

In conclusion, our analysis of the residual stream reveals that: 1) LLMs exhibit internal mechanisms for identifying conflicts, and this signal can be leveraged to detect conflicts effectively in the mid-layers of LLMs; 2) LLMs display distinct skewness patterns in the residual stream when using different sources of information, which provides insights on the model's behaviour.

5.3.1 Background and Methods

Residual Stream We examine the Transformer architecture from the perspective of the residual stream [175, 435]. In this framework, tokens flow through the model, with their embeddings being modified by vector additions from the attention and feed-

forward blocks in each layer. We denote the hidden states at position i at l-th layer as $\mathbf{h}_i^l \in \mathbb{R}^d$, where d is the dimension of the internal states of the model. The model produces the initial residual stream $\mathbf{h}_i^0$ by applying an embedding matrix to the tokens. Then, the model modifies the residual stream by a sequence of L layers Transformers, where each Transformer layer consists of a Self-Attention block and MLP at l-th layer. Formally, denote $\mathbf{a}_i^l$ and $\mathbf{m}_i^l$ as the activations of Self-Attention and MLP respectively, the update of the residual stream at l-th layer is $\mathbf{h}^{l'} = \text{LayerNorm}(\mathbf{h}^{l-1}) + \mathbf{a}_i^l$ and $\mathbf{h}^l = \text{LayerNorm}(\mathbf{h}^{l'}) + \mathbf{m}_i^l$.

Linear Probing Linear probing [116, 695, 11] is a commonly used technique to analyse whether certain information is encoded within the residual stream of a language model. Specifically, for an activation $\mathbf{x}$ from the residual stream, i.e., $\mathbf{h}$, $\mathbf{a}$, or $\mathbf{m}$, a logistic regression model is applied to perform binary classification: $P(y = 1|\mathbf{x}) = \delta(\mathbf{x}\mathbf{W})$, where $\mathbf{W} \in \mathbb{R}^{d\times 1}$ is the learned weight that linearly projects the activation into a scalar value , and δ is the Sigmoid function that outputs the likelihood of probed information existing in the activation.

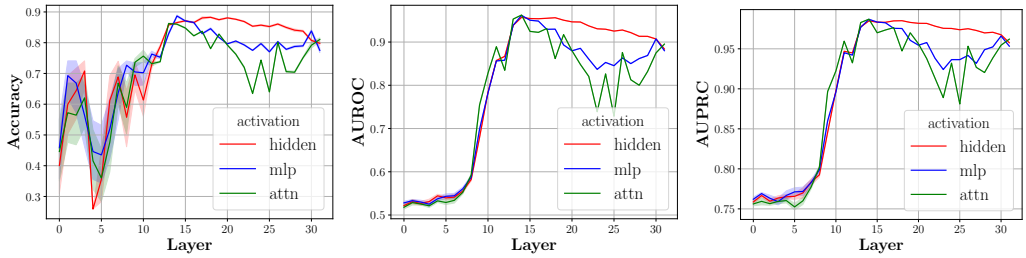
5.3.2 Experimental Setup

Problem Setup Following previous studies [372, 255, 626, 554, 599], we use open-domain question-answering (ODQA) tasks to investigate the behaviours of LLMs when there is a conflict between the model's parametric knowledge and contextual knowledge. In ODQA datasets with knowledge conflicts, each instance is presented as (q, e_M, e_C, a_M, a_C), where q is the question, e_M is the evidence that supports the memorised knowledge, e_C is the evidence that conflicts with the language model's memorised knowledge, a_M is the answer based on and e_M, and a_C is the answer based on the e_C. The model's parametric knowledge is tested in the close-book setting, where the model generates answer a_M based on the question q without external evidence. We generate the answers using a greedy decoding strategy. We use three in-context demonstrations to align the answer format and, for fairness, use the same in-context demonstrations in all experiments.

Datasets and Models We use NQSwap [372], Macnoise [255] and ConflictQA [626] to analyse the residual stream when knowledge conflicts arise. We present the experiment results of NQSwap using Llama3-8B [399] in the main paper, and the results of other datasets and models are provided in Section 5.3.6 and Section 5.3.7. The training details of the probing model are presented in Section 5.3.5

5.3.3 Results and Findings

In this work, we aim to answer the two following research questions: *1)* Can we identify the conflict between context and parameter knowledge by probing the residual stream? *2)* Can we know which source of knowledge the models will use before they generate the answers? We probe and analyse the residual stream to answer these two questions in the following parts.



(a) Accuracy **(b)** AUROC **(c)** AUPRC

Figure 5.31. Accuracy, AUROC, and AUPRC of probing models on detecting the knowledge conflicts based on the activations of Llama3-8B. The probing results on hidden state, MLP and Self-Attention activation are coloured red, blue and green, respectively. More analysis is presented in Section 5.3.6.

Identifying Knowledge Conflicts by Probing the Residual Stream We analyse whether language models can identify contextual-parametric knowledge conflicts by probing the residual stream. To this end, we create two groups of instances, $D^{e_C} = \{(q, e_C)\}$ and $D^{e_M} = \{(q, e_M)\}$, where the model generates answers based on conflict evidence in D^{e_C} and non-conflict evidence in D^{e_M}. The probing model is trained to classify whether a given activation is from D^{e_C} or D^{e_M}. We probe the residual stream at the final position to determine if the model is aware of the conflict during the first token generation. This is because the hidden state at the last position in the output layer is used to predict the first token of the answer. For each activation $\mathbf{h}^l$, $\mathbf{a}^l$ and $\mathbf{m}^l$ at each layer, we train a probing model to classify whether it belongs to D^{e_M} or D^{e_C}.

As shown in Fig. 5.31(b) and Fig. 5.31(c), the AUROC and AUPRC of the probing models increase from the first layer to the 14th layer, and this trend is same across the hidden state, MLP, and Self-Attention activations. In Fig. 5.31(a), the accuracy of the probing models at the early layers is random; similar to the trend of AUROC and AUPRC, the accuracy also reaches the highest score at the 14th layer. The above observation indicates that the residual stream does not contain information about knowledge conflict at the early layers. This information rises from around the 8th layer and reaches the highest point at the 14th layer.

After the 14th layer, the probing model's performance decreases slightly until the last layer. Besides, we also observe that the probing results of MLP and Self-Attention activations show a significantly lower accuracy than the hidden state after the 14th layer, which may suggest that MLP and Self-Attention do not provide further conflicting information into the residual stream. We find the same trend using Llama2-7B as shown in Fig. 5.34.

Analysis of the Residual Stream When LLMs Using Different Sources of Knowledge We investigate the distribution patterns of the residual stream when the language model uses different sources of information to generate the answer. Based on the model's predictions on instances belongs to D^{e_C}, we classify them into two groups: $D^{e_C}_{a_C}$ and $D^{e_C}_{a_M}$. Here, $D^{e_C}_{a_C}$ represents the set of instances where the model's predictions align with a_C, while $D^{e_C}_{a_M}$ contains the instances where the predictions align with a_M. The model uses contextual knowledge and parametric knowledge to answer the questions from $D^{e_C}_{a_C}$ and

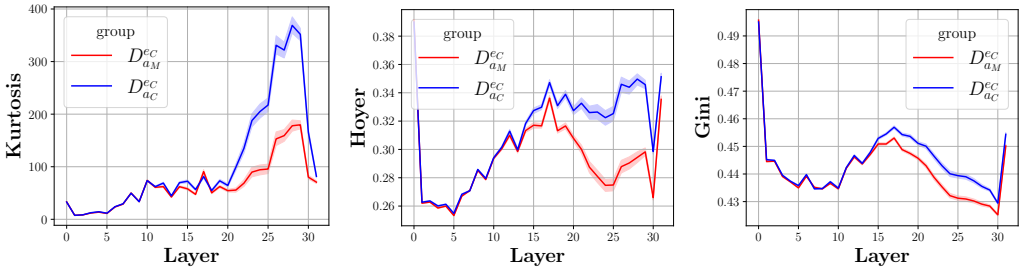


Figure 5.32. Skewness of the hidden state activations of Llama3-8B when in presence of knowledge conflicts. Blue and red lines represent the skewness of hidden states from $D^{e_C}_{a_C}$ and $D^{e_C}_{a_M}$, respectively. Higher scores indicate a more skewed distribution. Additional analyses are available in Section 5.3.7.

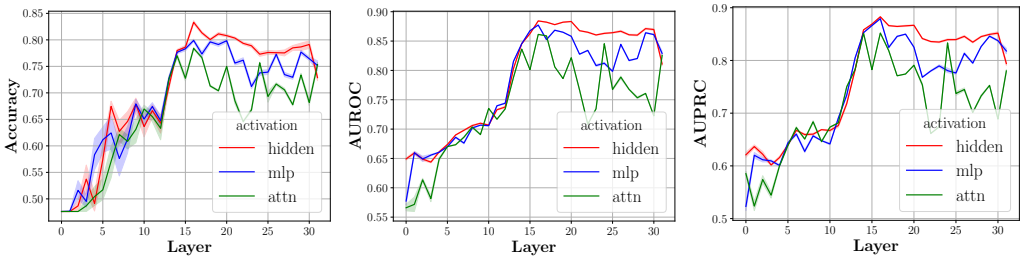


Figure 5.33. Accuracy, AUROC, and AUPRC of probing models on predicting which source of knowledge the model will use to predict the answer in Llama3-8B. More results are Skewness of the hidden state activations of Llama3-8B when the model uses knowledge from different sources to predict the answer. Additional results are available in Section 5.3.9.

$D^{e_C}_{a_M}$, respectively.

First, we examine the residual streams' distribution patterns in the two groups of instances $D^{e_C}_{a_C}$ and $D^{e_C}_{a_M}$. We measure the skewness of the residual stream using Kurtosis, Hoyer and Gini index. We present the results of NQSwap using Llama3-8B in Fig. 5.32, and more results are provided in the Section 5.3.7. We find that when the model uses contextual knowledge for prediction ($D^{e_C}_{a_C}$, blue lines shown in Fig. 5.32), the residual stream shows a significantly skewed distribution compared with using parametric knowledge from the 20th to 30th layers. Therefore, the distribution patterns of the residual stream can indicate the model will use different sources of knowledge. It provides the foundation for predicting the model's behaviour in advance, which can be used to mitigate the generation of undesirable responses in advance.

Based on the above observation, we probe the residual stream to analyse the possibility of predicting which source of knowledge will be used to generate the answer. The probing model is trained to classify whether the model will generate a_C or a_M based on the activation from $D^{e_C}_{a_C}$ or $D^{e_C}_{a_M}$. We present the probing results in Fig. 5.33. We observe that the probing model's performance gradually improves from the first layer to the 16th layer, which occurs after the signal of knowledge conflict has already reached its peak at the 13th and 14th layers. This observation suggests that the decision of which knowledge to use occurs after the detection of the knowledge conflict signal.

5.3.4 Related Work

Contextual and parametric knowledge conflict can happen when the retrieved external knowledge in the context does not agree with the parametric knowledge which is memorised during pre-training [372, 633, 626, 554, 599, 379]. Previous works found models may prefer the contextual knowledge [599, 554, 626, 442] when parametric and contextual knowledge conflicts, and the relevance, length, and the number of the evidence will influence the model's preferences [626, 554]. To detect the conflict, previous work [599] designed a multi-step prompting strategy to detect the knowledge, which involves parametric knowledge generation, fine-grained sentence consistency checking, and potential conflict reduction. However, this pipeline significantly reduces efficiency and lacks an understanding of the mechanism of how LLMs detect and resolve conflict.

5.3.5 Probing Model Training Settings

For all probing experiments, we train the probing model with an L_1 norm regularisation. The training objective is $\mathcal{L} = -\log P(y = y_i) + \lambda \|W\|_1$, where we set λ to 3×10^{-4} and y_i is the label. We train 20 times with different random seeds for each probing task, and we report the average and deviation in our experiments. We split the training and test datasets for the probing tasks, ensuring no overlapping questions between them.

5.3.6 More Experimental Results on Knowledge Conflict Probing

We present the knowledge conflict probing results on Macnoise, NQSwap, ConflictQA using Llama2-7B in Fig. 5.34, Fig. 5.35 and Fig. 5.36. The results match the trend discussed in Section 5.3.3, where the model exhibits an internal mechanism for identifying conflicts. The signal of knowledge conflict peaks around the 13th to 14th layers and gradually decreases in the later layers.

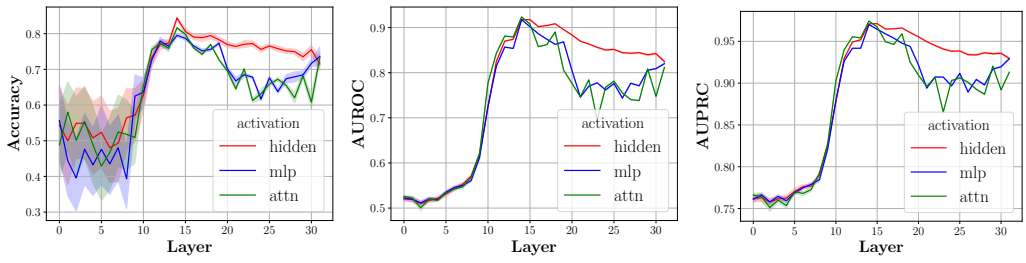


Figure 5.34. Knowledge conflict probing results using Llama2-7B on NQSwap.

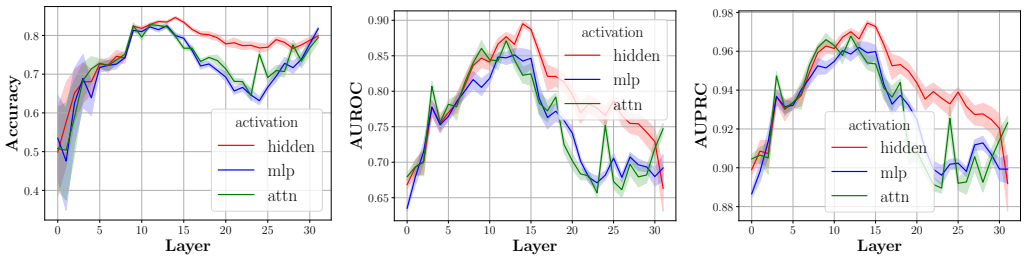


Figure 5.35. Knowledge conflict probing results using Llama2-7B on Macnoise.

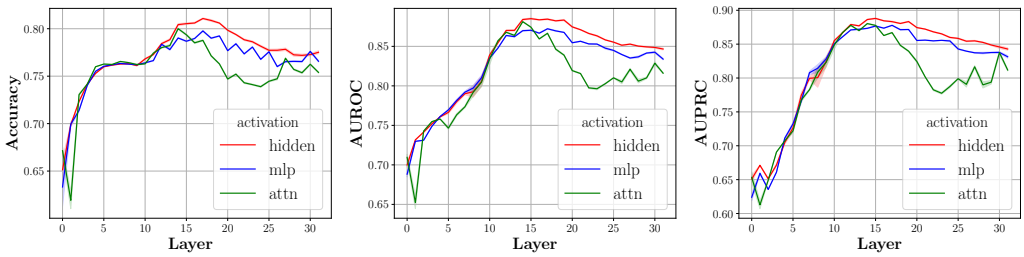


Figure 5.36. Knowledge conflict probing results using Llama2-7B on ConflictQA.

5.3.7 More Analysis of Skewness Patterns of Residual Streams

We present the skewness of the hidden state of Llama2-7B on NQSwap in Fig. 5.37. It shows the same pattern as we discussed in Fig. 5.32, where the residual stream exhibits significantly more skewed distribution when using contextual knowledge compared with using parametric knowledge from the 17th layer.

In addition to NQSwap, we analyse the skewness pattern using Macnoise [255] and ConflictQA [626]. As shown in Fig. 5.38, Fig. 5.39, Fig. 5.40, we find that the model also shows a similar skewness pattern with NQSwap, where the residual stream exhibits a more skewed distribution from middle layers when the model uses the contextual knowledge.

We also analyse the skewness of MLP and Self-Attention activations, presented in Fig. 5.41, Fig. 5.42, Fig. 5.43, and Fig. 5.44. However, we do not observe a specific skewness pattern in MLP and Self-Attention activations.

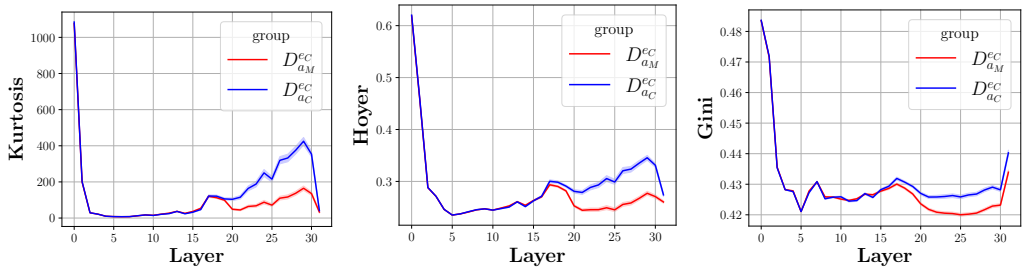


Figure 5.37. Skewness of the hidden states of Llama2-7B on NQSwap.

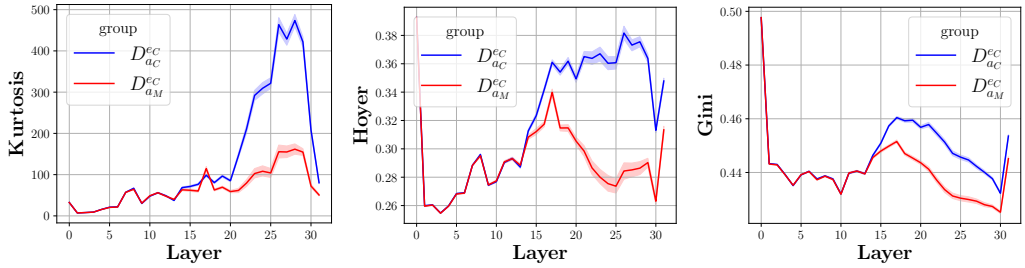


Figure 5.38. Skewness of the hidden states of Llama3-8B on Macnoise.

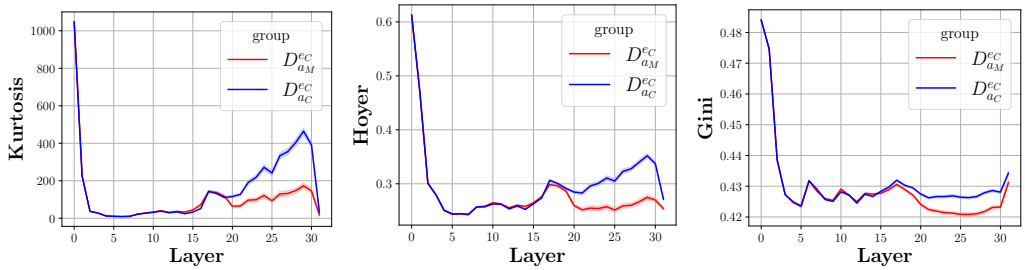


Figure 5.39. Skewness of the hidden states of Llama2-7B on Macnoise.

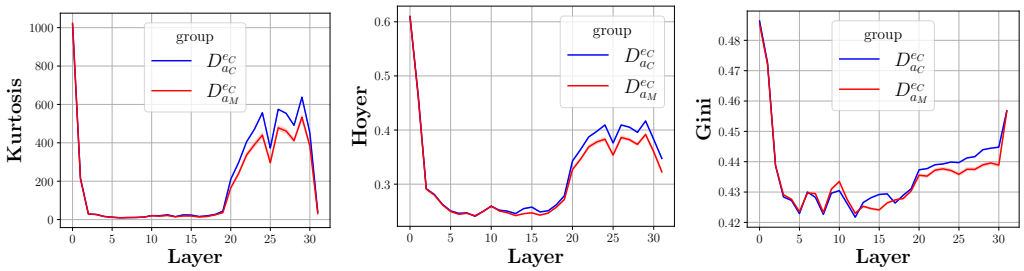


Figure 5.40. Skewness of the hidden states of Llama-27B on ConflictQA.

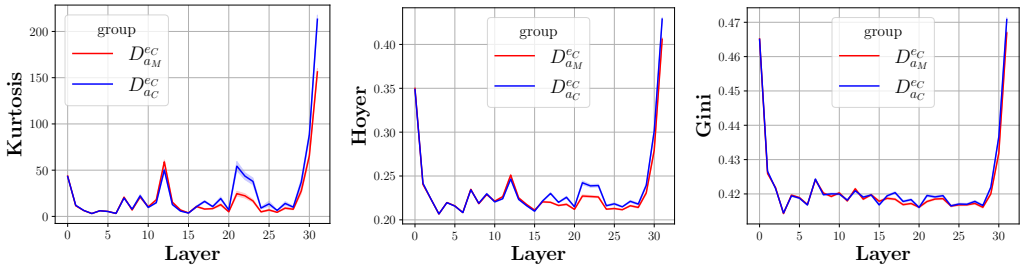


Figure 5.41. Skewness of the MLP activation of Llama3-8B on NQSwap.

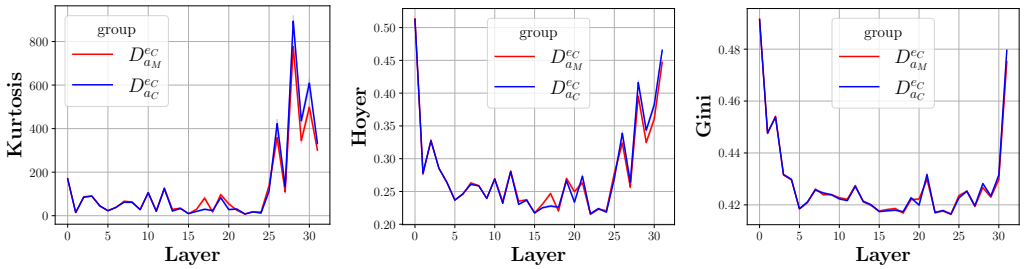


Figure 5.42. Skewness of the Self-Attention activation of Llama3-8B on NQSwap.

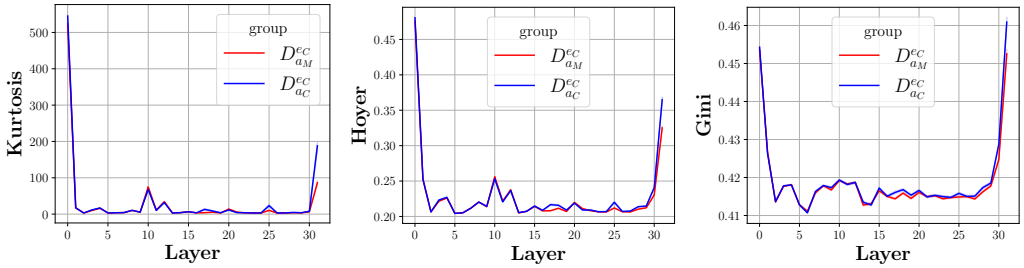


Figure 5.43. Skewness of the MLP activation of Llama2-7B on NQSwap.

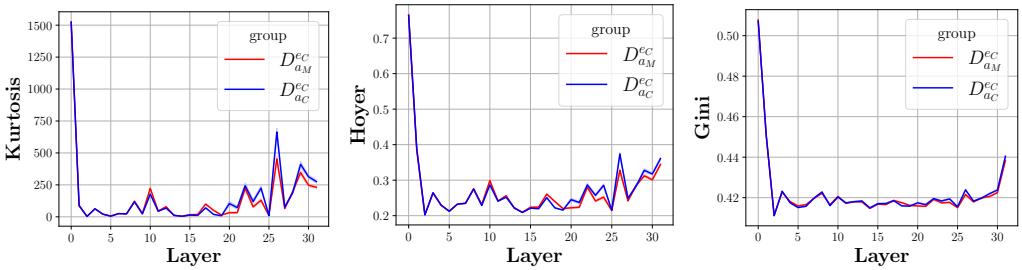


Figure 5.44. Skewness of the Self-Attention activation of Llama2-7B on NQSwap.

5.3.8 L1 Norm and L2 Norm Values of Residual Streams

We present L1 Norm and L2 Norm of the residual stream in the Fig. 5.45 and Fig. 5.46. We found that though the residual stream show distinct skewness patterns in $D_{a_C}^{e_C}$ and $D_{a_M}^{e_C}$, the L1 norm and L2 norm of the them do not have a significant difference.

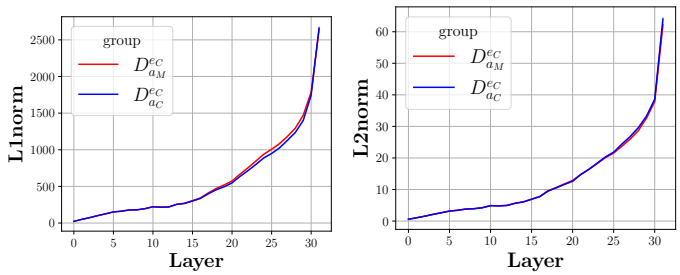


Figure 5.45. L1 norm and L2 norm of the hidden states of Llama3-8B on NQSwap.

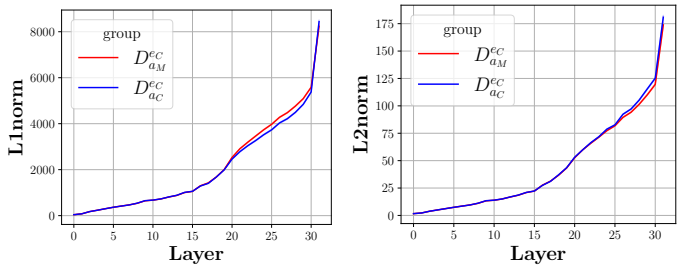


Figure 5.46. L1 norm and L2 norm of the hidden states of Llama2-7B on NQSwap.

5.3.9 More Experimental Results on Knowledge Selection Probing

We present additional knowledge selection probing results on NQSwap and Macnoise using Llama2-7B and Llama3-8B in Fig. 5.47, Fig. 5.48 and Fig. 5.49. The results show a similar trend as shown in Fig. 5.33, where the probing model reaches the highest accuracy at around the 17th layer, which is later than the aggregation of knowledge conflict signal at the 14th layer.

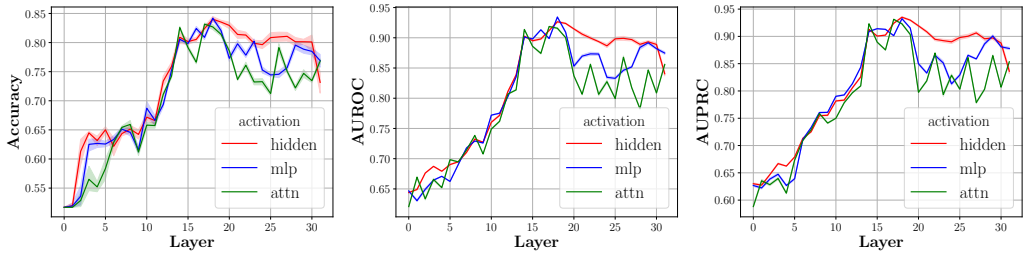


Figure 5.47. Knowledge selection probing results using Llama2-7B on NQSwap.

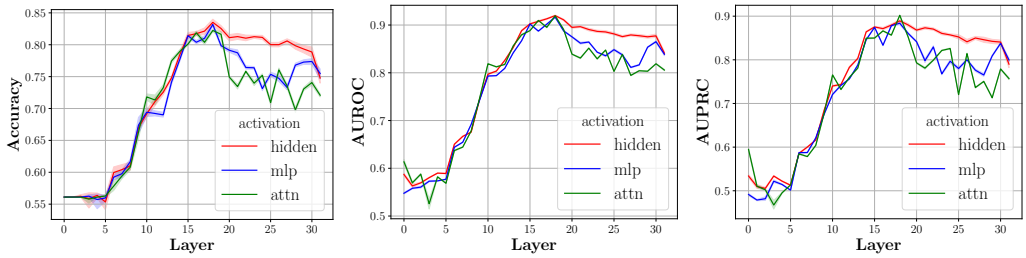


Figure 5.48. Knowledge selection probing results using Llama2-7B on Macnoise.

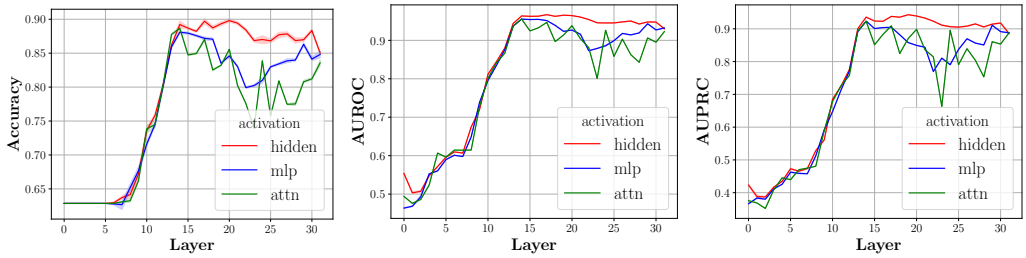


Figure 5.49. Knowledge selection probing results using Llama3-8B on Macnoise.

5.3.10 Conclusion

In this work, we analyse the residual stream of the language models when context-parameter knowledge conflicts. First, we find that LLMs exhibit internal mechanisms for identifying conflicts in the mid-layers. Second, we find that the residual stream shows distinct skewness patterns when the model uses context and parametric knowledge to predict. Our analysis provides insights into the behaviour of LLMs in the presence of knowledge conflicts.

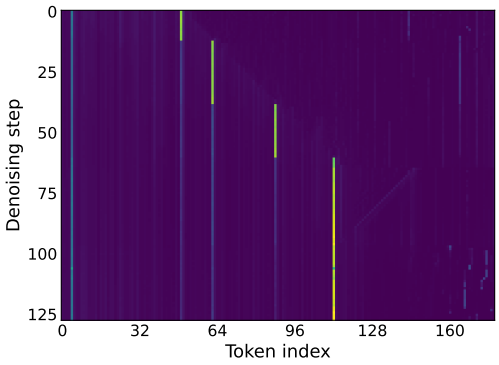


Figure 5.50. Incoming attention scores for each token in LLaDA-8B [427] across denoising steps. Unlike autoregressive models, DLMs exhibit attention sinks that shift across the sequence as tokens are progressively unmasked.

5.4 Attention Sinks in Diffusion Language Models

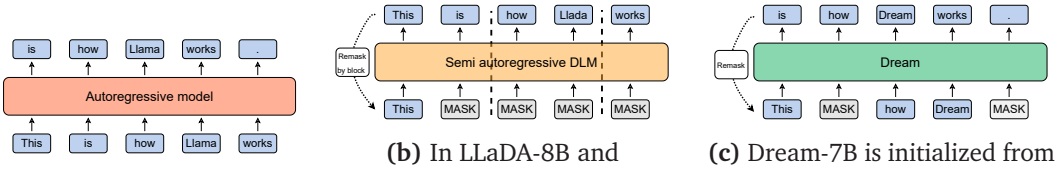
Large Language Models (LLMs) have driven a paradigm shift across numerous scientific and industrial domains, demonstrating remarkable capabilities in language understanding, generation, and reasoning [3, 18, 640, 398]. This rapid progress is rooted in the transformer architecture and the attention mechanism [586]. While attention is a critical aspect of the transformer's effectiveness, it also gives rise to complex and often non-intuitive emergent phenomena.

One of the most striking traits of these behaviours is the "attention sink" [622, 403]. This consists in the fact that, in most autoregressive models (ARMs), a small subset of tokens consistently receives a disproportionate amount of attention from other tokens in the sequence. The pattern is not limited to language, and similar patterns have been observed in Vision Transformers [134] and encoder-only transformers [508], suggesting it may be a fundamental property of attention-based deep networks.

Recently, masked Diffusion Language Models (DLMs) have emerged as an alternative to the dominant autoregressive paradigm [427, 650, 564, 320, 643, 600, 542, 511, 690, 369]. Unlike Autoregressive Models (ARMs), which generate text strictly from left to right, DLMs iteratively refine a fully masked sequence through successive denoising steps [427, 650, 643]. Generation is based on the unmasking of an initial fully masked sequence of tokens, that the model progressively "denoises" over multiple steps to produce a coherent fully unmasked output. Crucially, DLMs employ a bidirectional attention mechanism. While this bidirectional information flow is key to their parallel, non-causal generation process, the precise impact of this architecture on the inner workings of DLMs remains largely unexplored.

In this work, we present an empirical study of attention patterns in DLMs, focusing specifically on the attention sink phenomenon. We analyse three state-of-the-art open-source masked DLMs: Dream-7B [650], a model initialized from a pre-trained ARM; LLaDA-8B [427], a large-scale model trained from scratch; and MMaDA-8B [643], a multimodal DLM trained from LLaDA-8B. Our analysis reveals that DLMs do exhibit attention sinks, but these sinks possess unique dynamic properties rarely seen in their autoregressive counterparts.

Unlike the static attention sinks well-documented in ARMs, most of the sinks in



(a) In ARMs, each token is used to predict the next one in the sequence. Then, the predicted token is appended to the input sequence and fed again to the model, in a left to right fashion.

(b) In LLaDA-8B and MMaDA-8B a sequence of `[MASK]` tokens is passed as input to the model. The model then performs N denoising steps, gradually unmasking tokens inside each block, before proceding to the next block from left to right.

(c) Dream-7B is initialized from an ARM. Each token is used to predict the following one. At each inference step the entire input masked sequence is passed to the model. Unlike in other DLMs, Dream-7B uses the current token to predict the next one.

Figure 5.51. Snapshot of an inference step for different language models. ARMs and Dream-7B predict the next token, while MMaDA-8B and LLaDA-8B predict the current one. MMaDA-8B and LLaDA-8B perform semi-autoregressive block decoding, where only tokens in the current block are unmasked, while Dream-7B may unmask a token at any position.

DLMs are unstable and their position actively shifts across the iterative denoising process. Additionally, while ARMs are extremely sensitive to removing the sink tokens, we find that DLMs are significantly more robust to this intervention. We attribute this property to their decoding strategy that unmasks only the tokens with highest probabilities in the sequence, and the lack of a causal mask that limits the attention interaction among tokens. To summarize, our primary contributions are the following:

leftmargin=* We conduct an empirical study on attention patterns in DLMs, and provide empirical evidence that attention sinks consistently emerge in these models.

leftmargin=* We characterize the dynamic properties of these sinks, showing they can disappear and shift positions during inference, and we introduce a metric to track their intensity and location across denoising steps.

leftmargin=* We investigate how model performance is affected by removing sinks, and show DLMs are robust to sink masking.

5.4.1 Related Work

Diffusion Language Models

Language modelling has traditionally been dominated by autoregressive models that generate text sequentially, one token at a time. While this paradigm has proven highly successful, DLMs have emerged as an alternative, offering token generation through an iterative denoising processes with potential efficiency advantages [341, 613, 298, 369, 338, 612]. Some applications of diffusion to language modelling operate in continuous space, first embedding discrete tokens into continuous vectors, applying diffusion-based denoising, and then mapping back to discrete tokens [342, 552, 214, 163].

While theoretically elegant, this approach introduces additional complexity in handling the discrete nature of language. A more direct approach emerged with discrete diffusion models, which operate directly on token vocabularies [31, 212, 257, 84]. Starting from fully masked sequences of `[MASK]` tokens, these models iteratively predict and refine tokens through a process reminiscent of BERT-style masked language modelling [247, 213]. Several works [31, 247, 213] have adopted this paradigm but faced significant scaling challenges, remaining limited in size while autoregressive models scaled to billions of parameters.

Recently, discrete DLMs have gained traction thanks to open-source models like Dream-7B [650], MMaDA-8B [643] and LLaDA-8B [427, 690, 369], which have successfully scaled to 7 billion parameters and beyond, narrowing the performance gap with ARMs.

In this work, we investigate attention patterns in large discrete DLMs, that operate directly on the vocabulary space.

Attention Sink in Transformers

Attention Sink refers to the common phenomenon observed in transformers where a small subset of tokens consistently receives a disproportionate amount of attention from other tokens in the sequence. This behaviour was initially discovered in [622], and leveraged for efficiency. After this, other works have then explored the sink phenomenon, characterizing properties of sink tokens like high L_2 norm in the hidden state activations [556, 86] or low L_2 norm in the key projection [158, 227]. Similar properties have been also observed in the vision domain [134]. Several works have attempted to explain the emergence of attention sinks in transformers. [228] offers an empirical study of how attention sinks manifest in transformer models, specifically focusing on ARMs. [40, 41] and [450] investigate the phenomenon analytically and show how attention sinks act as a bias for ARMs and can mitigate information over-squashing. Finally, [508] analyses attention sinks from a geometric perspective, and shows that they emerge to establish stable coordinate systems in the model's high-dimensional latent space. While these works analyse sinks in both decoder and encoder transformers, we are the first to observe and investigate this phenomenon in the context of DLMs.

5.4.2 Background on Masked Discrete Diffusion

Traditional ARMs model the probability of a text sequence $\mathbf{x} = (x_1, x_2, \ldots, x_L)$ of length L by decomposing the joint probability into a product of conditional probabilities, generated in a strict, left-to-right order [281, 48]. This decomposition is given by:

$$p(\mathbf{x}) = p(x_1) \prod_{i=2}^{L} p(x_i | x_1, \ldots, x_{i-1}) \tag{5.14}$$

where x_i is the token at position i, and $p(x_i|x_1, \ldots, x_{i-1})$ is the probability of the current token conditioned only on all preceding tokens. Masked discrete DLMs offer a non-autoregressive, parallel alternative. Instead of generating tokens one by one, they model a Markov diffusion process over discrete token sequences. This consists of two complementary phases: a fixed **forward corruption process** and a learned **reverse**

denoising process. The forward process systematically corrupts a clean data sequence $\mathbf{x}_0$ (the original text) over a series of time steps $t \in [0, T]$ by progressively replacing tokens with a special mask token [MASK]. Starting with the clean sequence $\mathbf{x}_0$, a noisy sequence $\mathbf{x}_t$ at time step t is generated by a Markov transition $q(\mathbf{x}_t|\mathbf{x}_{t-1})$. The marginal distribution of a token $\mathbf{x}_t^i$ at time t conditioned on its clean version $\mathbf{x}_0^i$ is defined by a masking schedule $\alpha_t \in [0, 1]$. The complete forward process is the joint distribution over all intermediate noisy states, a product of the Markov transitions:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{5.15}$$

In the denoising process, a model $p_{\boldsymbol{\theta}}$, parametrized by θ, reverses this noising process, generating new data from a fully masked sequence $\mathbf{x}_T$ back to a clean sequence $\mathbf{x}_0$. More specifically, reverse transition $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is parameterized by the model, which is trained to estimate the true reverse conditional probability $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$.

In practice, the model $p_{\boldsymbol{\theta}}$ is often trained to predict the clean data $\mathbf{x}_0$ from the noisy input $\mathbf{x}_t$ at a given time t, and this prediction is then used to approximate the reverse transition. The model output is a distribution over the original tokens, from which the next, less-noisy state $\mathbf{x}_{t-1}$ is sampled.

In this work we consider three masked discrete DLMs: LLaDA-8B [427], MMaDA-8B [643] and Dream-7B [650]. LLaDA-8B and MMaDA-8B are trained from scratch, with a masked language modelling loss where a token x_i is masked during the forward process, and the model learns to predict the token itself ($x_i \to$ [MASK] $\to x_i$). At inference time, LLaDA-8B and MMaDA-8B use semi-autoregressive block diffusion, where the input sequence is divided into blocks, and the model gradually unmasks all tokens inside the corresponding block in a left-to-right manner [24], (see Figure 5.51). Dream-7B, on the other hand, is initialized from an autoregressive model to leverage the pretrained weights and its training objective employs a "shift operation" [650, 213]. More specifically, when a token x_i is masked, Dream-7B is trained to predict x_{i+1}, similarly to an autoregressive model ($x_i \to$ [MASK] $\to x_{i+1}$). In Figure 5.51 we provide a comparison and visual explanation of how the different types of inference are implemented.

5.4.3 Analysis of Attention Sinks in Masked Diffusion Language Models

Previous work has shown that attention sinks emerge in most transformer-based architectures, regardless of the data domain and training strategy [228, 508, 622, 134]. Attention sinks are characterized by the disproportionate attention score they receive from all the tokens in the sequence, and can be easily identified as vertical bright lines in attention maps (like the one we show in Figure 5.50). To validate the presence of attention sinks in DLMs, we first analyse the distribution of attention scores in LLaDA-8B and show it in Figure 5.52. We see that only a few tokens, the sinks, capture a very high attention score consistently. Similar patterns emerge for Dream-7B and MMaDA-8B (see Section 5.4.7). We now define a metric to characterize and locate attention sinks in DLMs.

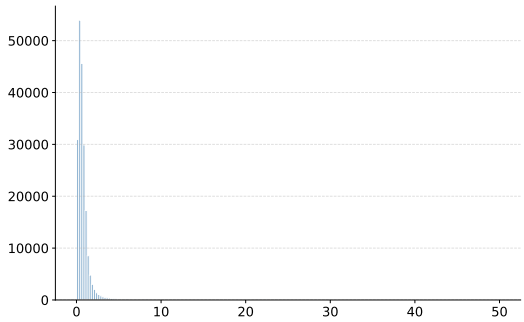
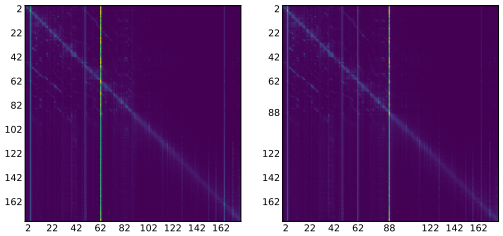
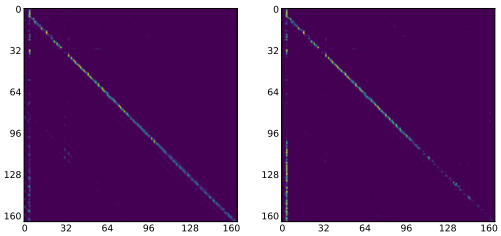


Figure 5.52. Distribution of attention scores in LLaDA-8B [427] across denoising steps. Only a few tokens, the attention sinks, receive a very high attention score, while the majority of tokens in the sequence have scores close to zero.



(a) Moving sink in LLaDA-8B. Attention plots at step 38 (Left) and step 39 (Right). The sink shifts from position 62 to 88 after one denoising step.



(b) Moving sink in MMaDA-8B. Attention at step 36 and step 37 . Observe that this sink absorbs the self-attention from each of the tokens paying it attention.

Figure 5.53. Moving sink in LLaDA-8B, and MMaDA-8B.

Definition of Attention Sink

Consider an encoder-only transformer model. For a single attention head h and layer l, we have that the attention score is defined as:

$$A_{ij} = \text{softmax}_j\left(\frac{q_i^\top k_j}{\sqrt{d}}\right)$$

where q_i and k_j are the query and key projections for token i an j respectively, and A_{ij} represents the amount of attention that token i pays to token j. In a DLM attention is bidirectional, and we obtain a distribution of attention scores across the entire sequence *at each denoising step*. Given the attention scores, we define the cumulative attention score for a token j as the average attention it receives from all tokens in a specific denoising step t:

$$\bar{A}_j^{(t,l,h)} = \frac{1}{S}\sum_{i=1}^{S} A_{ij}^{(t,l,h)}$$

where S is the sequence length, and $A_{i,j}^{(t,l,h)}$ represents the attention score from token i to token j at denoising step t, in head h of layer l. We then identify attention sinks as tokens that receive a cumulative attention score substantially larger than the average.

Attention Sink. We formally define a token j at a specific denoising step t, in head h of layer l to be a **sink token**, if its cumulative attention score exceeds the average cumulative attention score of all other tokens by at least a threshold ϵ:

$$j \text{ is a sink token if } \bar{A}_j^{(l,h)} > \frac{1}{S-1}\sum_{k\neq j}\bar{A}_l^{(l,h)} + \epsilon \tag{5.16}$$

This definition ensures that sink tokens represent significant outliers in the attention distribution. In all our experiments we use $\epsilon = 3$, which we selected to filter out at least the 96% of tokens in sequence, and empirically showed a sufficient robustness to detect sinks while also serving as a filter for tokens that did not exhibit a sink characteristic. We further discuss the value of ϵ in Section 5.4.8.

Sink Patterns

Our analysis reveals that DLMs exhibit distinct types of attention sinks with unique dynamic properties not observed in ARMs. We find that sinks do not necessarily appear in the beginning of the sentence, but also show up in the middle or towards the end, which is possible as attention in DLMs is bidirectional. Along with the typical static sink that is frequently observed in ARMs, we identify a new kind of attention sinks that we call **moving sinks**.

Moving sinks appear at different positions during denoising and exhibit widely different patterns according to layer depth and backbone model. Moving sinks are not consistent across diffusion steps, i.e. they do not remain at the same position across all diffusion steps and may move or even vanish throughout the denoising process. We show an example in Figure 5.53a. We now analyse how attention sinks appear in the considered pre-trained models.

LLaDA-8B exhibits diverse moving sink patterns with consistency across different sequences. Moving sinks often remain at a specific position for some consecutive denoising steps, before vanishing. Nonetheless, we also find some edge cases in which the moving sinks behave extremely unstably, as we see in Figure 5.55b, where a sink appears for only one timestep before vanishing on the next one.

As we progress to deeper layers, the number of sinks decreases, converging to one or two sinks per layer, as we show in Figure 5.54. The deepest layers showcase a particular type of moving sinks, where masked and unmasked tokens maintain separate attention sinks, and switch gradually. We show an example of this phenomenon in Figure 5.55a. Notably, LLaDA-8B demonstrates a strong semantic basis for sink selection as sinks consistently form on punctuation marks (periods, commas), whitespace, and end-of-sequence tokens. This pattern suggests that LLaDA-8B, trained from scratch as a diffusion model, developed semantically-aware attention mechanisms that identify structurally important tokens as reference points for attention.

Dream-7B showcases a sink behaviour that follows primarily a positional rather than a semantic pattern. Unlike LLaDA-8B, Dream-7B's sinks often originate at the rightmost masked token and shift leftward as tokens are progressively unmasked, regardless of the token content, as we show in Figure 5.57b. This right-to-left migration is most prominent

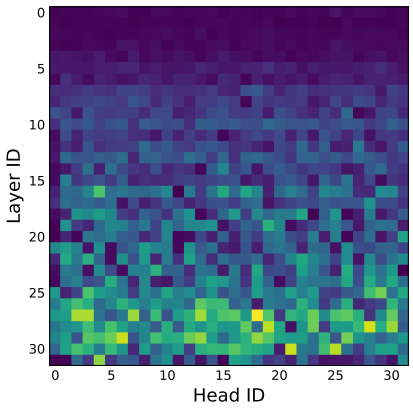
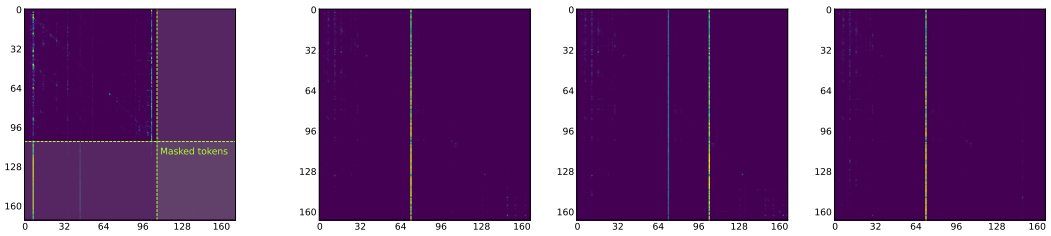


Figure 5.54. Cumulative attention score for LLaDA-8B's sink across heads and layers. The variation of the model's main sink token is displayed across the different heads and layers, averaged through time. Brighter colours indicate higher attention score. In later layers there are usually fewer sinks and the attention score is therefore higher, as it is shared among fewer sink tokens.



(a) **(b)**

Figure 5.55. Different types of moving sinks in LLaDA-8B. (a) A particular kind of moving sink in which attention is split according to token type. Some heads exhibit this behaviour in which the masked tokens heavily attend to a specific sink, while the unmasked ones are more concentrated on another one. This heatmap is from step 32, at the precise end of a block, explaining why we have a perfect line separating all the unmasked and masked tokens. **(b)** A sink appears at step 96 but suddenly disappears at step 97.

in early layers and creates a dynamic attention flow that follows the unmasking frontier. This positional nature of Dream-7B's sinks likely stems from its initialization from a pre-trained autoregressive model. The inherited representations may be less refined for bidirectional attention, causing the model to rely on positional cues rather than semantic content for sink formation. Dream-7B's positional bias represents a difference from LLaDA-8B's semantic approach and suggests that initialization strategy and positional embeddings significantly influences attention organization in diffusion models [508].

MMaDA-8B presents the most stable sink behaviour among the three models, with sinks that are generally static and less frequent. When sinks do manifest they often remain fixed at their initial positions throughout the entire generation process, as we show in Figure 5.57a. The model exhibits minimal moving sinks, with most layers showing no clear sink patterns at all. This stability contrasts with the dynamic patterns

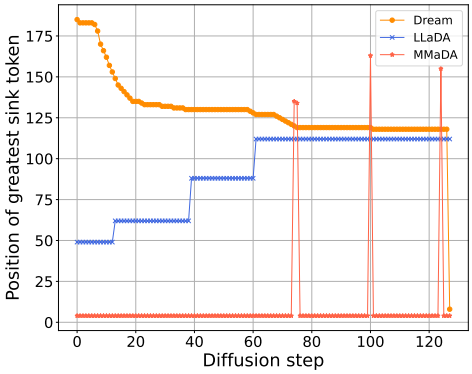
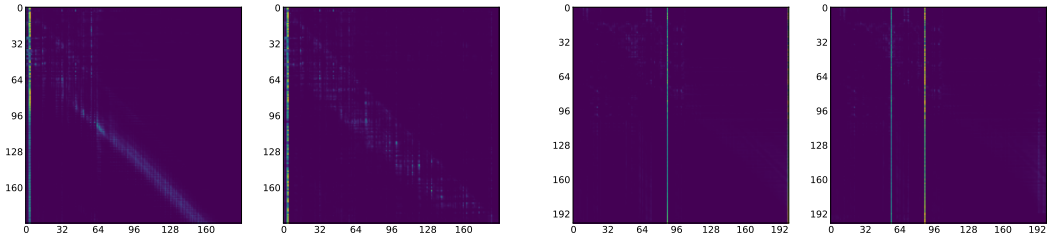


Figure 5.56. Example of how sinks move over time. The largest sink from each model's specific heads is selected at each iteration. See how the attention shifts according to the explained phenomena. Note that these are sinks for a specific head of the model and not the actual averaged one.



(a) Fixed sink in MMaDA-8B. MMaDA-8B often exhibits a static sink at the beginning of the sequence. In different denoising steps, the sink stays consistently at the beginning of the sequence.

(b) Moving sinks in Dream-7B typically shift from right to left. The sink moving is on step 32 (Left) and at the rightmost position. While at step 33 (Right) the sink has moved towards the centre.

Figure 5.57. Fixed sink in MMaDA-8B and moving sink in Dream-7B.

in LLaDA-8B and Dream-7B, potentially reflecting MMaDA-8B's different multimodal training data. The static nature of MMaDA-8B's sinks more closely resembles traditional autoregressive models, though the bidirectional attention mechanism still allows for unique patterns not possible in causal models. For instance, in Figure 5.53b we show that a considerable amount of tokens shift their attention towards an already unmasked token from one step to the other.

In Figure 5.56 we show how sinks behave in different models. We select a specific head from each model and compare the position of the largest sink detected by our metric. We observe that while MMaDA-8B exhibits a mostly static sinking behaviour, sinks tend to shift position in Dream-7B and LLaDA-8B. More specifically, we observe that in LLaDA-8B the sink tends to shift right as more blocks are denoised, while it moves from right to left in Dream-7B.

Robustness of DLMs to Masking Sinks

Previous studies have demonstrated that attention sinks play a crucial role in transformer-based models, with their removal typically causing catastrophic performance degra-

Dataset	Sinks	DREAM-7B [650]	LLADA-8B [427]	MMADA-8B [643]	LLAMA-3.1-8B [399]
GSM8K	Unmasked	$0.82_{\pm 0.01}$	$0.76_{\pm 0.01}$	$0.54_{\pm 0.01}$	$0.85_{\pm 0.01}$
	Masked ϵ_0	$0.79_{\pm 0.01}$	$0.75_{\pm 0.01}$	$0.53_{\pm 0.01}$	$0.02_{\pm 0.00}$
	Masked ϵ_1	$0.78_{\pm 0.01}$	$0.73_{\pm 0.01}$	$0.54_{\pm 0.01}$	$0.02_{\pm 0.00}$
	Masked ϵ_2	$0.75_{\pm 0.01}$	$0.55_{\pm 0.01}$	$0.37_{\pm 0.01}$	$0.01_{\pm 0.03}$
HumanEval	Unmasked	$0.60_{\pm 0.03}$	$0.37_{\pm 0.03}$	$0.16_{\pm 0.02}$	$0.66_{\pm 0.04}$
	Masked ϵ_0	$0.64_{\pm 0.03}$	$0.37_{\pm 0.03}$	$0.16_{\pm 0.03}$	$0.00_{\pm 0.00}$
	Masked ϵ_1	$0.61_{\pm 0.03}$	$0.39_{\pm 0.03}$	$0.18_{\pm 0.03}$	$0.00_{\pm 0.00}$
	Masked ϵ_2	$0.57_{\pm 0.03}$	$0.35_{\pm 0.03}$	$0.09_{\pm 0.02}$	$0.00_{\pm 0.00}$

Table 5.8. Performance of DLMs and ARMs under attention sink masking. Thresholds ϵ_0, ϵ_1, and ϵ_2 correspond to masking 1, 5, and 10 top-ranked attention sinks, respectively. While LLama-3.1-8B exhibits severe degradation when masking a single sink token, LLaDA-8B, Dream-7B, and MMaDA-8B maintain competitive performance across masking settings, suggesting that parallel inference in DLMs provides robustness to attention sink removal.

dation [622, 228, 40]. However, given that attention sinks in DLMs exhibit markedly different and more dynamic patterns compared to ARMs, we investigate whether DLMs demonstrate similar sensitivity to sink masking during generation.

We evaluate the three DLM variants — LLaDA-8B, Dream-7B, and MMaDA-8B — on both coding and mathematical reasoning tasks using the GSM8K [115] and HumanEval [97] datasets. GSM8K contains grade-school level math word problems, while HumanEval comprises programming problems designed to evaluate code generation and reasoning capabilities. For each model, we conduct two sets of evaluations: (1) using the original, unmodified model, and (2) masking attention scores directed toward the top-K attention sinks identified by our metric (Equation 5.16). We vary the threshold parameter ϵ, where smaller values result in masking a larger proportion of sinks. Specifically, we select ϵ_0, ϵ_1 and ϵ_2 to mask the top $1, 5$ and 10 sinks respectively.

Surprisingly, the tested DLMs exhibit only modest performance degradation when sinks are masked (Table 5.8). For all the tested DLMs, masking one sink leads to a degradation in performance smaller than 1%. Substantial degradation occurs only when ϵ is decreased further to mask 10 sinks, and mostly in MMaDA-8B. In contrast, applying the same masking procedure to LLama-3.1-8B results in severe performance drops even when masking a single sink token, confirming prior findings that ARMs are highly sensitive to attention sink removal [622, 228].

We hypothesize that this increased robustness stems from the parallel inference mechanism inherent to DLMs, which may provide alternative attention pathways when primary sinks are unavailable. We explore this hypothesis further in Section 5.4.4.

Implementation details. We evaluate our models in PyTorch [455] using the checkpoints released on Hugging Face transformers [584] and the official `lm evaluation harness` scripts [195]. We use the same hyper-parameters specified in the respective original papers. For LLaDA-8B, we use a block size of 32 and a generation length of 256 tokens for GSM8K and 512 for HumanEval. For Dream-7B, which does not use semi-autoregressive block generation, we adjust only the generation length and diffusion step parameters according to the original settings. We successfully reproduce the reported results for LLaDA-8B, Dream-7B, and LLama-3.1-8B using these configurations. However, we were

unable to reproduce the original results for MMaDA-8B despite following the published implementation details, and we therefore report our own evaluation results for this model. Throughout our analysis, we employ $\epsilon = 3$ for sink detection, a threshold that empirically balances robust sink identification with the exclusion of non-sink tokens.

5.4.4 Discussion

Dynamic Sinks and Positional Encoding

Recent work on encoder-only models notes that attention sinks can shift usually around special markers like `[CLS]` or `[EOS]` and connects this behaviour to the use of absolute positional embeddings [508]. However, we find that DLMs, despite using Rotary Positional Embeddings (RoPE, **(author?)** 553), show extremely varied and dynamic sink patterns, including sinks that move and others that split attention between masked and unmasked tokens. These appear all over the text sequence, often on important structural tokens (like punctuation). The emergence of sink tokens on semantic markers suggests that the sinking behaviour is driven not only by the positional encoding or token index in the sequence [508, 41], but also by training dynamics and frequency of the token in the training corpus [556, 322].

Robustness to Masking Sinks

A notable result from Section 5.4.3 is that DLMs keep working, although with a drop in performance, even when we mask their attention sinks, which would cause an ARM to fail completely. We believe this robustness comes from the bidirectional attention and the iterative denoising process working together to create stability that ARMs lack. In ARMs, attention is causal, and the sink token is usually a single, static anchor, that all future tokens rely on. The next token to predict is therefore usually highly dependent on the sink, and cutting its attention score causes the model to fail.

However, the bidirectional attention in DLMs lets every token see the full context at every denoising step. Additionally, at each step all tokens are considered for unmasking, and only the ones with highest probability (i.e., where the model is most confident) are actually unmasked. This iterative denoising process might ensure higher stability: when a sink is masked, the model likely becomes less confident about those tokens that are highly affected by the sink, and therefore not consider them for unmasking.

Long Context Modelling

In ARMs, attention sinks have been proven to act as a tool to control over-mixing and avoid representation collapse, especially in long contexts [41, 161]. However, attention sinks in ARMs are usually present only at the beginning of the sequence and represent a single point of reference for the entire generation. In contrast, DLMs offer a flexible inference and their sinks often shift position during generation. By dynamically directing attention to tokens that are currently most important for the ongoing prediction, DLMs might be able to maintain strong, long-range connections more effectively than ARMs that rely on a single, fixed bottleneck for information. Having the ability to access sinks at the end of the sequence might represent and advantage for long reasoning and

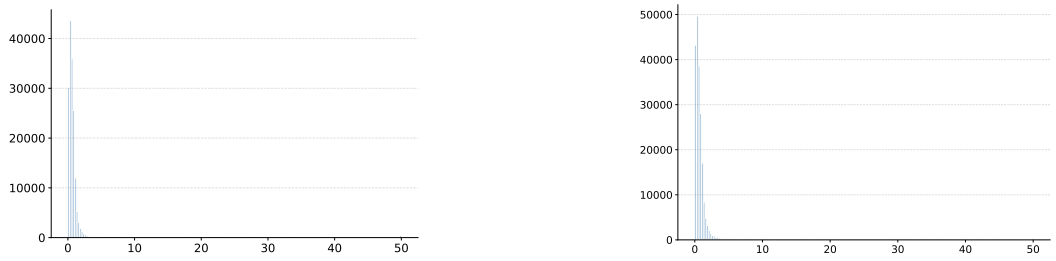


Figure 5.58. Distribution of attention scores in Dream-7B and MMaDA-8B

planning tasks [648, 647, 649], where the model needs a reference anchor in the future instead of the usual static one at the beginning of the sequence.

Additionally, for very long context generation in real-world deployment scenarios, sinks represent a single point of weakness. When the context exceeds the available GPU memory, the oldest part, typically including the `[BOS]` token, must be discarded. However, discarding sinks in ARMs has been shown to be catastrophic for downstream performance. DLMs on the other hand mitigate this limitation. Their moving sinks, which often appear in the future relative to the current generation step, allow the model to discard the past context without significant performance degradation.

5.4.5 Future Work

While our empirical analysis offers a general overview of sink behaviour in DLMs, it also raises several open questions. First, it remains unclear what type of information the model stores in the sinks that correspond to future positions. A promising direction to investigate this would be a mechanistic analysis, for instance using the Logit Lens [431].

Second, it is worth exploring whether sinks could be exploited for acceleration or compression, similar to their original use case in [622].

Finally, although we observed several sink behaviours (e.g., Figure 5.55a), we did not attempt to provide a detailed explanation of these phenomena. While such an investigation would be valuable, it would require an interpretability-focused study, which lies beyond the scope of this primarily empirical work.

5.4.6 Limitations

While we conducted an extensive study across three DLMs, our analysis is limited to instruct models, as we did not perform experiments on their corresponding base versions. Furthermore, we focused on attention sinks in pre-trained models and did not explore how modifications to the training procedure might influence their behaviour, an aspect that has recently been investigated for ARMs by [403, 436].

5.4.7 Additional plots

In Figure 5.58 we show additional plots of attention score distribution, displaying how a only a few tokens, the sinks, receive a disproportionate high attention score.

5.4.8 Selection of Sink Threshold

In Equation 5.16 we defined ϵ to be the threshold for classifying a token as a sink. In Figure 5.59 we show how the value of ϵ affects sink selection. We see that all the analysed DLMs filter out at least 96% of tokens when usigng $\epsilon = 3$.

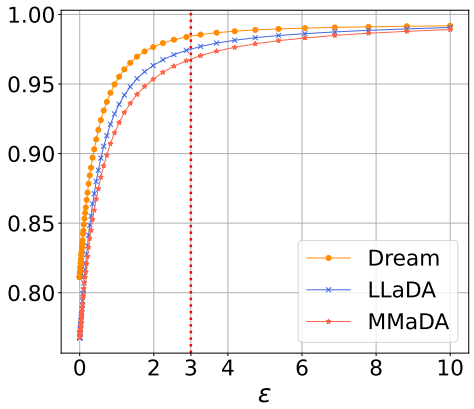


Figure 5.59. Percentage of tokens not considered as sinks when increasing the value of ϵ, for a sequence of 64 tokens. A balanced threshold is found at $\epsilon = 3$, which we used in this investigation to define that a token is a sink.

5.4.9 Conclusion

We presented the first empirical analysis of attention sinks in Diffusion Language Models, showing that they consistently emerge but behave differently from those in autoregressive models. In DLMs, sinks are dynamic, often shifting across denoising steps and aligning with semantic or structural tokens rather than fixed positions. Moreover, DLMs remain remarkably robust to sink masking, suggesting that their bidirectional and iterative generation distributes attention more evenly and avoids reliance on single anchor tokens. These findings reveal that diffusion models organize attention through flexible mechanisms, offering new insights into their internal dynamics and interpretability.

5.5 Interpretable Classification of Levantine Ceramic Thin Sections via Neural Networks

The archaeological context of the Levantine region is both particularly significant and complex, as numerous archaeological sites emerged during the Early Bronze Age, a period characterized by urbanization, trade networks, and political complexity [547, 221]. During this phase, the first urban societies in the region took shape. The process of urbanization was marked by the aggregation of local settlements, the formation of regional centers, and the fortification of villages, reflecting a shift toward more centralized form of organization. These transformations included the simplification of material culture and the reorganization of craft production [220]. One example of this process is Tell el-Far'ah (North), with strong fortifications and standardized ceramic production suggesting the presence of large workshops and inter-community exchange. While urbanization began earlier in the southern areas of the Levantine region, northern parts followed this path a bit later with key sites like Ebla showing signs of complex administration and organized settlement planning [603, 108].Therefore, over the past decades, there has been a growing interest in studying this region to understand and reconstruct the changes and developments that have been observed in various archaeological sites. The study of ancient ceramics plays a central role in understanding the social and cultural identity of the region from the Bronze Age, as it provides information on social, cultural, and technological development as well as the evolution of ancient civilizations. Indeed, an archaeometric characterization of ancient ceramic materials can define the nature and provenance of raw materials used in production that help to define trade routes and interaction among past societies [484, 441, 655].

Currently, petrographic analysis of ceramic thin sections is one of the most effective and widely used methods for archaeometric characterization and reconstruction of the technological and material features of ceramics. The petrographic analysis also includes the identification of *fabrics*, groups of thin sections sharing similar characteristics, and representative of the *chaîne opératoire*, i.e. the actions and technological choices that led to the formation of the final product [605, 607]. Therefore, the grouping of thin sections in *fabrics* has been widely applied to define the different "recipes" used in ceramic production and to distinguish between local or imported wares [606]. Specifically, petrographic analyses can help in the recognition of specific ware types as locals and in identifying patterns of exchange of pottery within the region [216, 222, 217, 127]. In other cases, they have contributed to the identification of relationships between ancient production centers and their surrounding geological settings [382, 33, 579]. However, reconstructing ancient exchanges and defining trade profiles remain challenging due to the still limited petrographic studies on large ceramic assemblages, resulting in a lack of comparative data useful for identifying potential relationships and interactions. Moreover, although minero-petrographic characterization is a well-established and effective method for studying pottery assemblages, it is time-consuming and challenging to compare results across different archaeological contexts. This is partly due to the dispersion of petrographic data across various sources such as specialist journals, monographs, conference proceedings, and excavation reports.

In recent decades, there has been a growing interest in applying Machine Learn-

ing (ML) and/or Deep Learning (DL) techniques to archaeology in a variety of ways, including the identification of archaeological sites and structures [89, 236, 577], and the classification or recognition of archaeological artifacts. Recent applications have involved the use of neural networks for the prediction of morphological features of lithic artifacts even when these are fragmented or incomplete [578, 430]. Other applications include the use of Laser-Induced Breakdown Spectroscopy (LIBS) combined with neural networks to attribute archaeological bone remains to specific individuals [539]. Additionally, Lithic Use-Wear Analysis (LUWA) using microscopic images has been employed to study the working traces on lithic artifacts [666]. Concerning the study of ceramics, automated methodologies have become powerful tools for resolving classification tasks, significantly contributing to the recognition of specific compositional, technological, or stylistic patterns [54, 458]. Today, several algorithms have been tested for the classification of pottery based on the study of their typology, decoration, and shape [273, 420]. ML models have also been applied in provenance analysis, where ceramic samples have been classified through the elaboration of geochemical data [478, 507, 15]. Among these approaches, Artificial Neural Networks (ANNs) have been employed to determine ceramic provenance using LIBS spectra [494]. ANNs have further been used for the analysis of ceramic thin sections with the aim of facilitating the *fabric* classification, in combination with image analysis to identify inclusions and pores in thin sections by working with parameters such as color, shape, size, percentage of inclusions and porosity [20]. Recent studies focused on the development of a DL model based on Convolutional Neural Networks (CNNs) to classify ceramic thin sections into their respective petrographic *fabrics* [376, 377].

CNNs are specifically designed for image pattern recognition. They consist of a series of layers among which the convolutional ones play a fundamental role, allowing the detection of characteristic patterns meaningful for accurate classification [326].

Recently, Transformers-based models have emerged as a significant alternative to CNNs. Vision Transformers [165] (ViTs) operate using multi-head self-attention instead of convolution and process input images by dividing them into patches. This approach captures both local and global dependencies, facilitating the identification of relationships across the entire image [489]. Yang et al. have explored the combination of channel and self-attention mechanisms within CNN architectures to improve their discriminative power in the classification of Ming-Qing ceramics [355].

Despite being extremely accurate, modern DL methods are highly complex, often consisting of hundreds of layers and parameters, making it difficult to determine which features the model relies on for classification. As a consequence, predictions are usually difficult to explain. For this reason, such models are often considered 'black boxes', as they do not provide any direct explanation or insights about their predictions [90]. In order to tackle this limitation, Explainable Artificial Intelligence (XAI) emerged as a field aimed at studying the interpretability of ML models [683, 23]. Understanding what the model has learned allows for further validation of its accuracy, also ensuring that it is based on the correct variables during the training phase, allowing the identification of potential limitations and building trust in the model's performance [443, 411]. In the context of XAI, saliency maps and attention maps are visual tools that help to understand a model's decision [201]. To the best of our knowledge, neither ViTs nor explainability techniques have been applied to the ceramic petrographic *fabric* classification. The only similar study applied Grad-CAM (Gradient-based Class Activation Maps) to visualize the

Table 9. Summary of the selected ceramic thin sections.

Archaeological sites	Age	N. of samples
Tell el-Far'ah (North)	Early Bronze I - II	55
Khirbat al-Batrawy	Early Bronze III - IV	40
Khirbat Iskandar	Early Bronze IV	35
Bethlehem	Early Bronze IV - Iron Age IB-II	23
Ebla	Early Bronze IV - Middle Bronze I	18
Jericho	Early Bronze	7

key features used by a CNN model for petrographic classification, although explainability was not its main focus [377]. Other related works have employed Grad-CAM to interpret CNN predictions for the typological and/or chronological classification of ceramic sherds and vessels [458, 174, 355].

The present research focuses on the application of automated methodologies for the classification of Levantine ceramics into their petrographic *fabrics* using deep learning models, specifically Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs). The aim is to investigate the potential of these models in the field of petrographic studies of pottery. The performance of these models is compared to evaluate their effectiveness in classification tasks. Furthermore, the research provides a partial interpretation of CNNs and ViTs using several forms of visual explanations. The objective is a deeper understanding of the classification process by visualizing the distinctive features that most influenced the models' predictions.

5.5.1 Materials and Methods

The starting point for the implementation of the DL models consisted of a phase of selection and acquisition of a large number of ceramic thin section images, which have been used as training and testing datasets. A total of 178 ceramic samples were selected, primarly dating to the Bronze Age, with a smaller number from the Iron Age. These samples come from six different archaeological sites across the Levantine region: Bethlehem (West Bank), Tell el-Far'ah North (West Bank), Khirbat Iskandar (Jordan), Khirbat al-Batrawy (Jordan), Ebla (Syria) and Jericho (West Bank) as shown in Fig. 5.60 (see Table 9 for more detailed information on the samples). While most of the sites are located in the southern part of the region, the inclusion of Elba provides important comparative data from the north. All samples have already been described and classified in terms of their respective petrographic *fabrics*, as well as their archaeological context in previous papers

Acquisition of thin section images was performed using the Axiocam 208 ZEISS camera connected to a ZEISS D-7082 Oberkochen petrographic microscope. For each ceramic section, we took multiple images under plane-polarized light (PPL) and cross-polarized light (XPL) at magnifications of 2.5X and 10X, resulting in a total of eight images per section to ensure the capture of important microstructural and mineralogical features. All images were compared and grouped into ten petrographic *fabrics*, taking

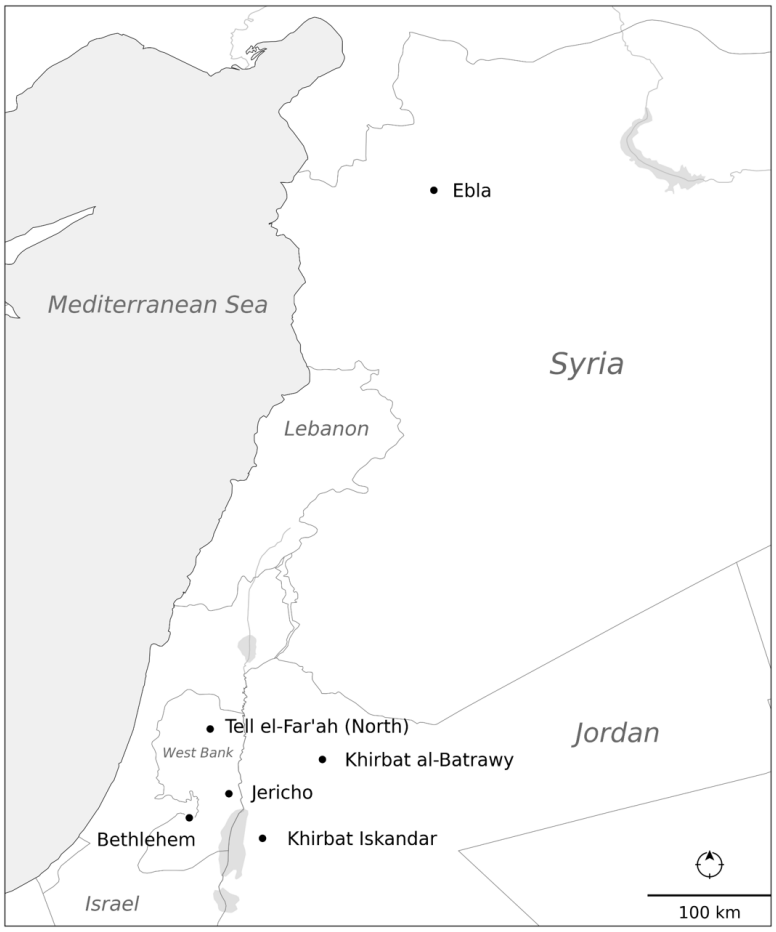


Figure 5.60. Map of the Levant showing the selected archaeological sites.

into account the results of the previous works. Samples from the different *fabrics* are presented in Fig. 5.61. The final dataset, consisting of 1,424 images grouped in 10 classes, is randomly split into training and testing sets with a ratio of 80:20. The larger portion was used for the training phase, while the smaller one was used to test the models on new data.

Implementation of CNN and ViT models

We select two types of deep learning models for this study: A CNN using the ResNet18 [245] architecture, and the more recent Vision Transformer [165]. CNNs are neural networks that process images through multiple convolutional layers, each detecting increasingly complex features. ResNet18 consists of 18 layers organized in 5 stages: an initial 7×7 convolutional layer followed by 8 blocks, each containing two 3×3 convolutional layers with batch normalization and ReLU activation. Each block includes a skip connection that adds the input to the block's output, helping to prevent vanishing gradients. The network concludes with an average pooling layer and a fully connected layer for classification.

Vision Transformers represent a fundamentally different approach: they first divide

input images into fixed-size patches (16×16 pixels in our case), linearly embed each patch, add positional embeddings, and process them through a series of transformer encoder blocks. Each encoder block contains a multi-head self-attention layer followed by a multi-layer perceptron (MLP). Our ViT implementation uses 12 transformer encoder blocks, each with 12 attention heads and a hidden dimension of 384.

We train our models to predict the class of a given image, minimizing a cross entropy loss function. To evaluate classification performance, we train the models and subsequently compare their performance on the held-out test dataset.

For both models, we use a transfer learning approach, fine-tuning pre-trained versions on the large ImageNet dataset [143]. This approach leverages the pre-training on ImageNet, a large dataset of 1.2 million images across 1,000 classes. We retain the pre-trained weights of the backbone layers, that have already learned to extract general visual features like edges, textures, and shapes, and only retrain the final classification layers to adapt to our specific classes. This transfer learning strategy allows the model to build upon previously learned features, making it more effective when working with limited data.

As part of our ablation study, we also evaluate a from-scratch approach, where all model parameters are randomly initialized and trained exclusively on our dataset.

We trained the ResNet-18 models for 60 epochs (from scratch) and 25 epochs (pre-trained) respectively. Both implementations used the AdamW optimizer [301] with learning rates and weight decay values of 3e-4. An adaptive learning rate scheduler was incorporated to dynamically adjust the learning rate throughout the training process. For the ViT models, training extended across 125 epochs (from scratch) and 60 epochs (pre-trained) using AdamW with a learning rate of 1e-4 and 1e-5 respectively, and a weight decay of 1e-4. To enhance model generalization and effectively expand our training dataset, we implemented various data augmentation techniques [645] including random axis flipping, color jittering, and random cropping. These augmentations were applied consistently during the training phase for both models to mitigate overfitting. We provide further details concerning the training hyper-parameters in 5.5.4.

5.5.2 Results

The mean and standard deviation of the metrics for CNNs and ViTs are summarized in Table 10. The reported values correspond to the average performance of the models initialized three times with different random seeds. For reference, we also show the results of the models trained from scratch without a pre-trained ImageNet checkpoint. As expected, the transfer learning approach led to a significantly higher performance. The ResNet18 model trained from scratch achieved an accuracy of 76.33 ± 0.58% after 60 epochs with a precision of 78.10 ± 0.73%, a result consistent with the small size of the dataset. In contrast, the pre-trained ResNet18, initialized with ImageNet weights, reached an accuracy of 92.11 ± 0.44% and a precision of 92.05 ± 1.35% in 25 epochs. ViTs are generally more effective for large datasets; as expected, the model trained from scratch underperformed, with an accuracy of 61.01 ± 0.33% and a precision of 58.74 ± 1.21% after 125 epochs. However, the pre-trained implementation performed well despite the small dataset, achieving an accuracy of 88.34 ± 0.29% and a precision of 88.44 ± 0.50% in 60 epochs, highlighting the effectiveness of transfer learning even with limited data.

Table 10. Mean and standard deviation of the accuracy, precision, recall (True positive rate) and F1-score (harmonic mean of precision and recall) metrics of the CNN and ViT.

Model Metrics	ResNet18 (Scratch)	Pre-trained ResNet18	ViT (Scratch)	Pre-trained ViT
Accuracy	76.33 ± 0.58%	92.11 ± 0.44%	61.01 ± 0.33%	88.34 ± 0.29%
Precision	78.10 ± 0.73%	92.05 ± 1.35%	58.74 ± 1.21%	88.44 ± 0.50%
Recall	72.16 ± 1.08%	90.15 ± 1.58%	59.93 ± 0.60%	86.60 ± 0.83%
F1-score	73.67 ± 0.71%	90.68 ± 1.54%	58.03 ± 0.47%	86.84 ± 0.70%

Fig. 5.62a presents the confusion matrix from the best run of the pre-trained ResNet18 model, showing the correlation between predicted and actual values and highlighting the number of correctly classified samples. The adjacent plot illustrates the evolution of training and testing accuracy over the epochs, allowing us to evaluate the learning progress of the model and identify potential overfitting or underfitting issues. Similarly Fig. 5.62b visualizes the best run results of the ViT model, presenting its confusion matrix along with the training and testing accuracy trends over a total of 60 epochs. It is important to note that the accuracy on the test set was monitored for visualization purposes only. The final evaluation of each model was based on its performance on the test set after training was completed.

Analysis of misclassified samples

In this section we perform an in-depth analysis of the confusion matrix of the models to verify whether the most difficult samples to assign to specific petrographic *fabrics* correspond to those misclassified by the models and also whether these samples are consistently misclassified, identifying potential biases or problematic samples. From the analysis of the confusion matrix of both the models (Fig. 5.62), it emerges that class 2 has the highest number of misclassifications with class 3. Class 2 represents the carbonate A *fabric*, characterized by the presence of microfossils in the matrix, while the main inclusions are carbonate rock fragments, making it very similar to the carbonate B *fabric* (class 3). The main difference between them is that carbonate B *fabric* contains an abundant presence of quartz instead of microfossils [391]. Observing the misclassifications of the ResNet18 and ViT models (Fig. 5.63a and b), we note that the first three images of sample J3 are misclassified in both models, mainly being confused with class 3. It is interesting to highlight that this sample was particularly difficult to classify, as its matrix contains both microfossils and dispersed quartz. This combination of characteristics places it in an intermediate zone between carbonate A and carbonate B *fabrics*, making its classification more complex. These observations suggest that both the models can struggle with classes that share similar mineralogical characteristics, particularly when a sample exhibits features of multiple petrographic *fabrics*. This kind of ambiguity also makes classification difficult for human experts, suggesting that some misclassifications may reflect the complexity of the samples more than limitations in model performance.

5.5.3 Explainability analysis

Explainability Techniques

Given that our study involves two distinct neural network architectures, ResNet18 and ViT, we employ different explainability techniques suited to their respective structures.

For ResNet18, we apply the Guided Gradient-weighted Class Activation Mapping (Guided Grad-CAM) [527], a widely used interpretability technique for convolutional neural networks. Grad-CAM generates a coarse localization map by computing the gradient of the output with respect to the feature maps in the last convolutional layer, highlighting the most influential regions in the input image for the network's decision.

For the Vision Transformer, we employ attention map visualization to interpret model decisions. Unlike CNNs, which rely on local receptive fields, ViTs use self-attention mechanisms to establish long-range dependencies between tokens [297]. Each ViT layer consists of multiple attention heads, each learning distinct patterns of feature dependencies across the input sequence. By visualizing attention maps across these heads, we gain insights into how the model distributes its focus over the image. Attention maps are particularly informative for ViTs because they directly represent the learned relationships between different image patches. In early layers, attention heads often capture local structures, while deeper layers aggregate global contextual information. By analyzing these attention distributions, we can interpret whether the model attends to semantically meaningful regions, such as object boundaries or discriminative features, aligning with human intuition for visual reasoning.

Explainability of the models in classifying ceramic thin-section

Guided Grad-CAM was used to visually explain the predictions of the pre-trained CNN model, highlighting the important features considered for classification. This technique allows us to observe how the model effectively identified the minerals characteristic of the *fabrics* as representative for classification. The regions indicated by the Guided Grad-CAM as important correspond to the predominant minerals that lead to the discrimination and grouping of the ceramic samples according to their petrographic *fabrics*. Fig. 5.64 provides several significant examples, where the heatmaps allow for an immediate and clear visualization of the areas that positively contributed to the correct classification of the images. To analyze potential differences, we applied this method to both PPL and XPL images of the same samples. Identifying key features in PPL images is more challenging, as they appear less distinct compared to XPL images, making mineral discrimination more difficult. However, the results suggest that inclusions can still be distinguished from the matrix based on their shape and color. In XPL images, interference colors further help in distinguishing the main inclusions. Particularly interesting are the results for class 8. The images in Fig. 5.64d show how the heatmaps exclusively highlight the basalt fragments, distinguishing them from the other inclusions present in the thin section in both the PPL and XPL images. This demonstrates the neural network's ability to identify and learn specific characteristics of the corresponding petrographic *fabrics*. See Fig. B1 in 5.5.5 to visualize the Guided Grad-CAM results for the other classes.

To deeper investigate these results, we applied Guided Grad-CAM to a sequence of images of the same sample at different rotations to determinate which parameters most influenced the classification and whether rotation and birefringence effects had

any impact on the results. This analysis was conducted on calcite and quartz *fabric* samples. Fig. 5.65 shows the rotation sequence of a calcite *fabric* sample, showing its progressive rotation until some crystals reached extinction. Calcite, characterized by high birefringence, is particularly distinctive in optical analyses and is a perfect case study to determine which features are considered important for classification. The resulting heatmaps indicate that rotation does not affect the recognition of specific patterns for classification. On the contrary, they highlight how the interference colors of the calcite crystals, their shape and edges, and also those in extinction are key elements considered by the model for classification. Regarding the analysis conducted on the representative quartz *fabric* sample in Fig. 5.66, the heatmaps highlight quartz minerals independent of their size and/or brightness. Color appears to play a role in identification; in fact, the extincted quartz is not well evidenced, probably because the model is not able to discriminate it from the matrix as it is quite dark as well. Additionally, it is interesting to note that there are few carbonate rock fragments present in the sample that are not highlighted by the heatmaps. This result suggests that the model selectively recognizes quartz minerals without considering other inclusions, thus confirming the consistency of the classification process.

We use attention maps to visualize the classification predictions of the pre-trained ViT model, illustrating how it correlates different patches during the classification process. Fig. 5.67 shows attention maps of representative samples from classes 2, 5, 7, and 8, which correspond to carbonate A, dolomite B, shell, and basalt *fabrics*, respectively. Specifically, the figure presents attention maps from head 3 for class 2 and 5 and from head 5 for class 7 and 8 across different layers, allowing a better observation of the evolution of the attention over the layers of the ViT. To investigate potential differences in attention maps under different polarization conditions, we visualized the samples in both PPL and XPL. The results indicate that the attention maps remain consistent across both images, suggesting that the model's ability to focus on relevant mineralogical features is not significantly affected by polarization. Furthermore, the evolution of attention maps throughout the layers of the ViT appears consistent for each respective attention head. In the first layers, attention is more broadly distributed, while in deeper layers, it starts to focus on specific regions corresponding to the characteristic minerals of the petrographic *fabrics*.

The most challenging aspect of visualizing the important areas in this case is that the inclusions are distributed throughout the entire thin section. This makes it more difficult for the model to focus on specific features, especially considering that ViTs operate globally by analyzing patches rather than individual pixels. For this reason, it is particularly interesting to emphasize the results of classes 5 and 7, which contain a high number of inclusions dispersed in the matrix. In both cases, the attention maps show that the model was still able to focus on specific characteristics of the respective *fabrics*. In class 5, the model mainly highlights some well-defined dolomite crystals that stand out clearly from the matrix, as well as the silicate rock fragment that is characteristic of this class. A similar pattern can be observed for the shell *fabric*, where the attention maps seem to emphasize the areas corresponding to the elongated shell fragments rather than the more common carbonate fragments. We report the attention maps of representative samples from the other classes in 5.5.5, Fig. B2.

Both the saliency and attention maps clearly highlight the characteristic minerals of the petrographic *fabrics* as key features for classification. This indicates that the

models distinguished the ceramic thin sections based on their inclusions, similar to the way traditional petrographic analysis groups ceramics by studying pores, inclusions, and the matrix. Although our models focused mainly on inclusions, it is interesting to note a certain correlation between deep learning-based classification and conventional petrographic methods.

5.5.4 Details on Model Architectures and Training

This section presents the architectural details and training configurations for both the ResNet-18 and DeiT-Small models used in our experiments. Hyperparameter selection was carried out based on the results of a 3-fold cross-validation performed on the training portion of the dataset (80% of the full set). The best average validation results guided the final choice of hyperparameters. The explored hyperparameters are reported in tables A3 and A6.

Table 11. ResNet-18 Architecture Hyperparameters.

Parameter	Value
Total layers	18
Convolutional layers	17 + 1 fully connected
Residual blocks	8
Input resolution	224×224
Parameters	$\sim$ 11.7M

Table 12. ResNet-18 Training Setup.

Hyperparameter	Value
Learning rate	3e-4
Weight decay	3e-4
Training epochs (scratch)	60
Training epochs (pre-trained)	25
Batch size (scratch)	20
Batch size (pre-trained)	32
Optimizer	AdamW
Learning rate scheduler	ReduceLROnPlateau

Table 13. ResNet18 hyperparameters explored during 3-fold cross-validation, for both pre-trained and from-scratch models.

Hyperparameter	Values Tested
Learning rate	{3e-4, 1e-4}
Weight decay	{3e-4, 1e-4, 1e-5}
Optimizer	{AdamW}

Table 14. DeiT-Small Architecture Hyperparameters.

Parameter	Value
Layers (Transformer blocks)	12
Heads	6
Hidden dimension	384
MLP dimension	1536
Patch size	16×16
Input resolution	224×224
Parameters	$\sim$ 22M

Table 15. Vision Transformer Training Setup.

Hyperparameter	Value
Learning rate (from scratch)	1e-4
Learning rate (pre-trained)	1e-5
Weight decay	1e-4
Training epochs (scratch)	125
Training epochs (pre-trained)	60
Batch size	32
Optimizer	AdamW
Learning rate scheduler	CosineLR

Table 16. DeiT-Small hyperparameters explored during 3-fold cross-validation, for both pre-trained and from-scratch models.

Hyperparameter	Values Tested
Learning rate	{1e-4, 1e-5}
Weight decay	{1e-4, 1e-5}
Optimizer	{AdamW}

5.5.5 Guided Grad-CAM and Attention Maps of Other Classes

In this section, we present the results of Guided Grad-CAM and attention maps for the remaining classes that were not shown in the main text of the paper.

5.5.6 Conclusion

This study demonstrated the potential of deep learning methodologies for the automated classification of ceramic thin sections based on their petrographic *fabrics*, expanding on previous research by Lyons et al. [376, 377]. The application of Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs) yielded high classification accuracy and enabled the grouping of thin sections with similar petrographic characteristics. Furthermore, explainability techniques, such as Guided Grad-CAM for CNNs and attention

maps for ViTs, provided insights into the models' decision-making processes, confirming their ability to recognize key mineralogical features relevant for *fabric* classification.

One possible limitation arises from the use of saliency maps, such as Guided Grad-CAM, which, while effective in highlighting important image regions, may not always provide a complete or precise explanation of model decisions. To address these challenges, future research could explore more recent explainability techniques, such as sparse autoencoders [192]. Furthermore, integrating multimodal data sources, such as geochemical analysis alongside petrographic imaging, could improve classification accuracy and robustness, opening the possibility for more applications in the field of minero-petrographic studies. For instance, this approach could contribute to ceramic provenance research by aiding in the identification of raw material sources in relation to the geological context of surrounding areas.

A key future step is expanding the dataset of ceramic thin section images to improve model generalization and performance across a broader range of *fabrics* and archaeological contexts. One current limitation is that models trained on a predefined set of known classes may struggle to generalize to previously unseen samples, particularly given the high degree of ceramic variability in the Levantine region, constraining their robustness in real-world applications. As a future step, it will be essential to evaluate the model on additional sets of thin-section images, distinct from those used during training. This will allow for a more realistic assessment of the generalization capabilities of the model, including its ability to identify samples that do not match any of the known classes (i.e., out-of-distribution detection). Moreover, recent advances in unsupervised learning and few-shot learning offer promising tools to address this challenge, potentially expanding the applicability of deep learning to archaeological scenarios.

Overall, this study highlights the potential of explainable AI in archaeometric research, providing a reproducible and efficient approach to ceramic analysis. Continued advancements in AI interpretability will be crucial for refining these methodologies, ensuring transparency, and facilitating broader adoption in archaeological and materials science applications.

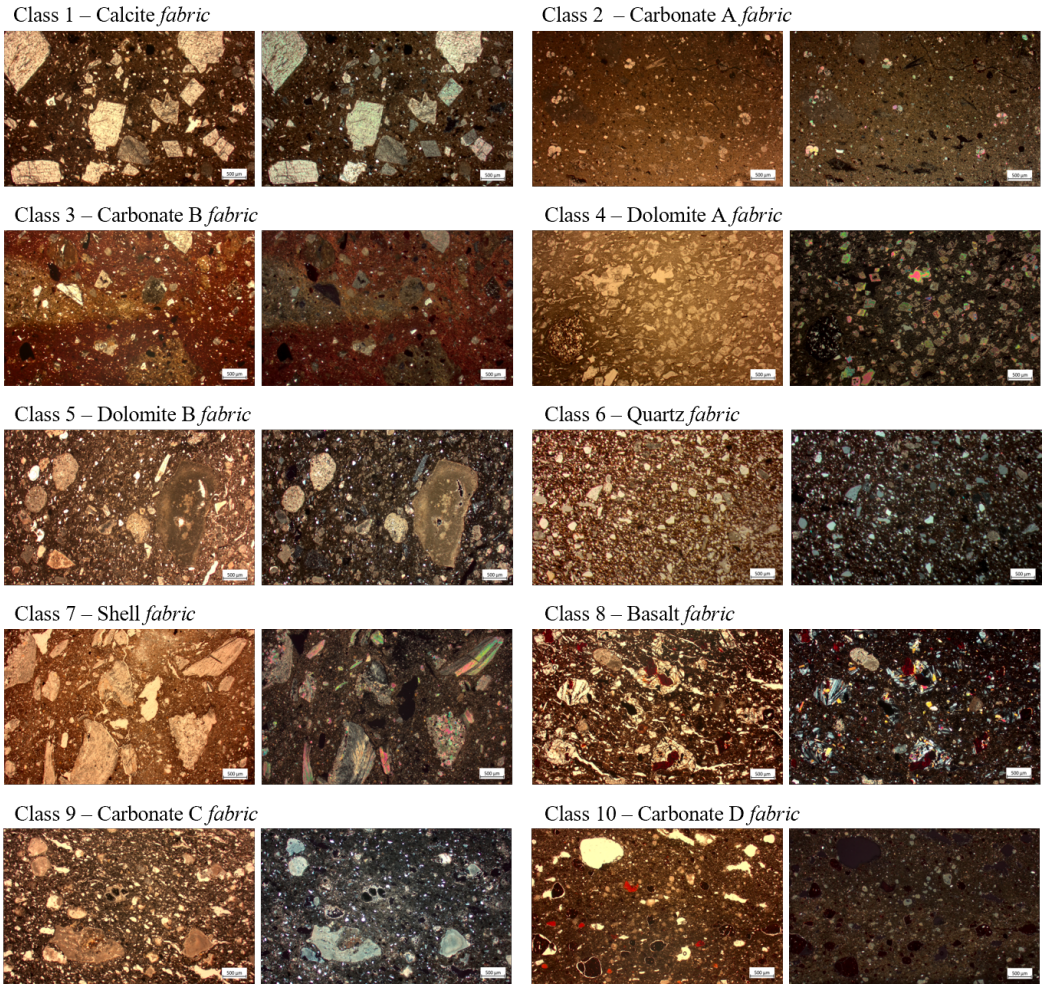


Figure 5.61. Thin section images of representative samples of the ten petrographic *fabrics* (2.5X, both in PPL and XPL respectively). Class 1 (sample T53): Characterized by the predominance of calcite inclusions; Class 2 (sample TF6): Predominance of fragments of sedimentary calcareous rocks, presence of microfossils in the matrix; Class 3 (sample TFN39): Fragments of sedimentary calcareous rocks, presence of quartz in the matrix; Class 4 (sample B1/14): Predominance of dolomite crystal inclusions; Class 5 (sample K7): Presence of dolomite, calcareous and siliceous rock fragments as dominant inclusions; Class 6 (sample TFMcamp1): Presence of predominant quartz inclusions; Class 7 (sample K12): Predominant presence of elongated bivalve shells and calcareous rock fragments; Class 8 (sample E467/3): Dominant presence of volcanic rock fragments identified as basalt; Class 9 (sample K11): Characterized by the dominant presence of calcareous and siliceous rock fragments; Class 10 (sample KB314/113): Predominance of calcareous fragments and iron oxide nodules. Complete description of these *fabrics* in [391, 389, 71, 72, 512, 388, 38, 390].

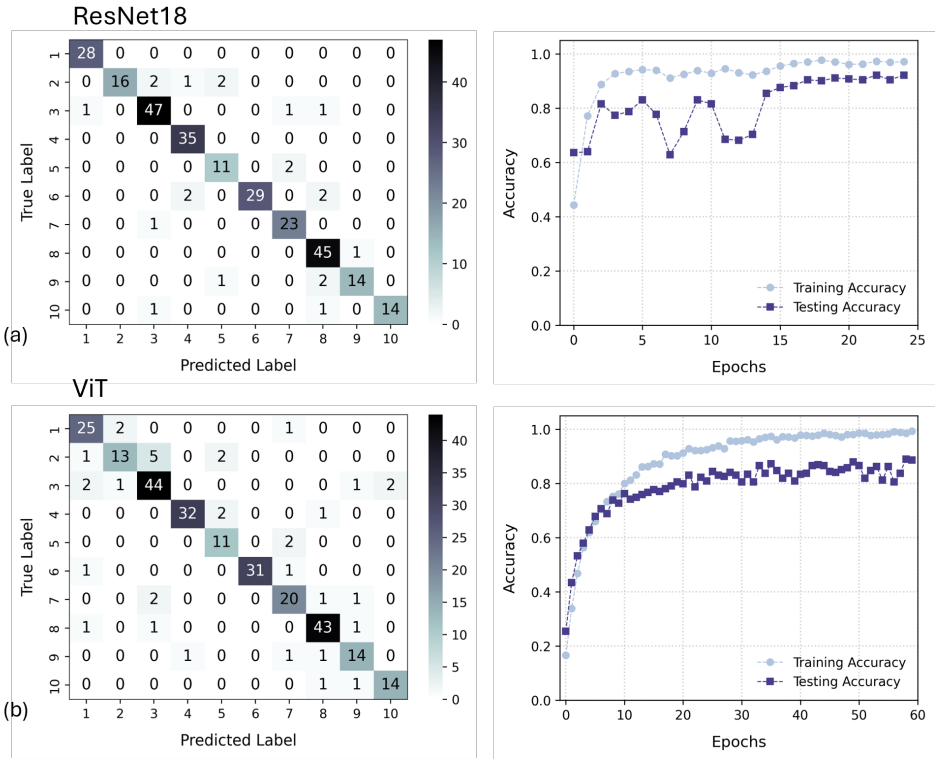


Figure 5.62. Classification performance of ResNet18 and Vision Transformer (ViT). (a) Confusion matrix and accuracy trends for ResNet18. (b) Confusion matrix and accuracy trends for ViT.

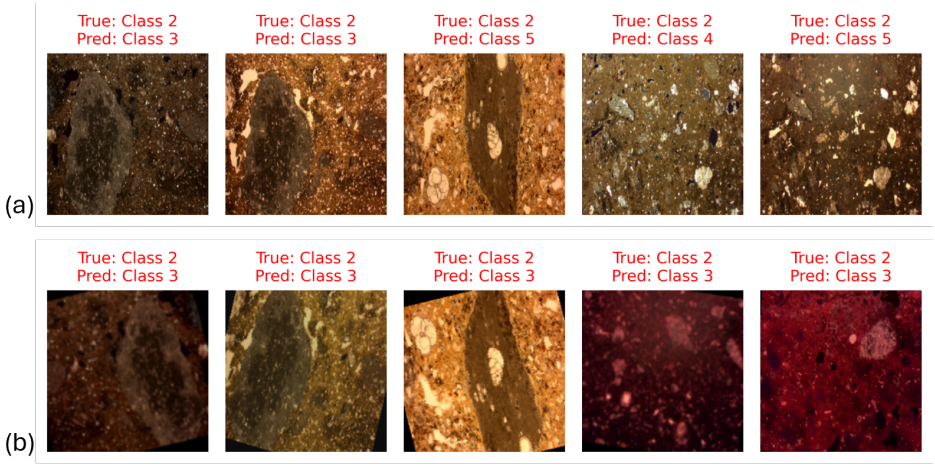


Figure 5.63. Comparison of ResNet18 and ViT misclassified predictions for class 2, representative of carbonate A *fabric*. (a) Images of ResNet18 misclassified predictions; (b) Images of ViT misclassified predictions.

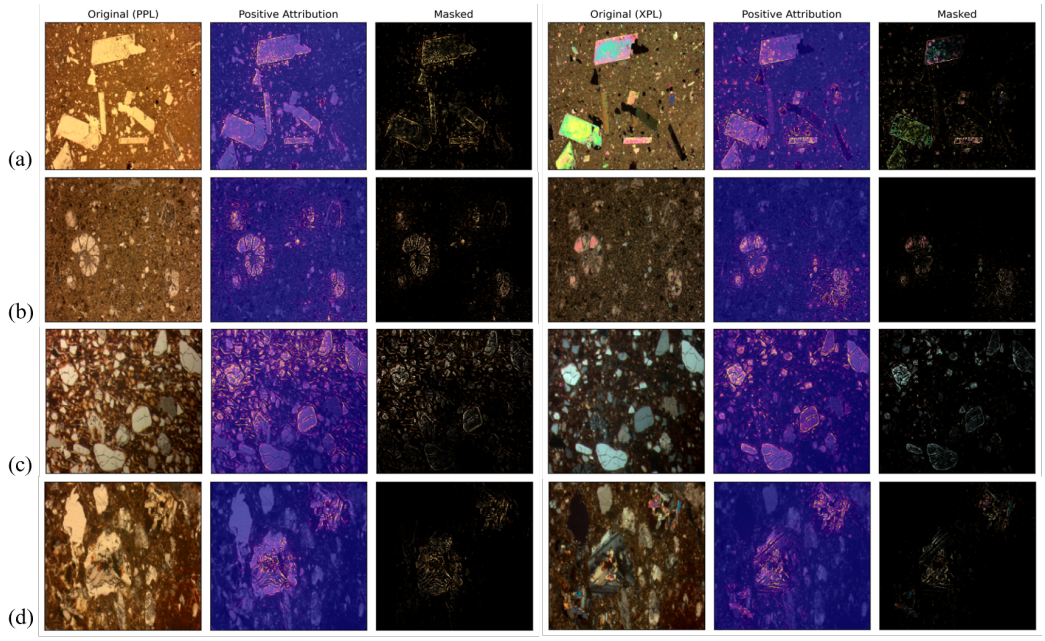


Figure 5.64. Guided Grad-CAM applied to pre-trained ResNet18. The figure shows the original images, the heatmaps overlapped to the thin-section images, and the masked images highlighting only the areas considered important for classification. (a) Class 1 (sample TF4): Calcite *fabric*; (b) Class 2 (sample TF6): Carbonate A *fabric*; (c) Class 6 (sample TFMcamp1): Quartz *fabric*; (d) Class 8 (sample E16/2): Basalt *fabric*.

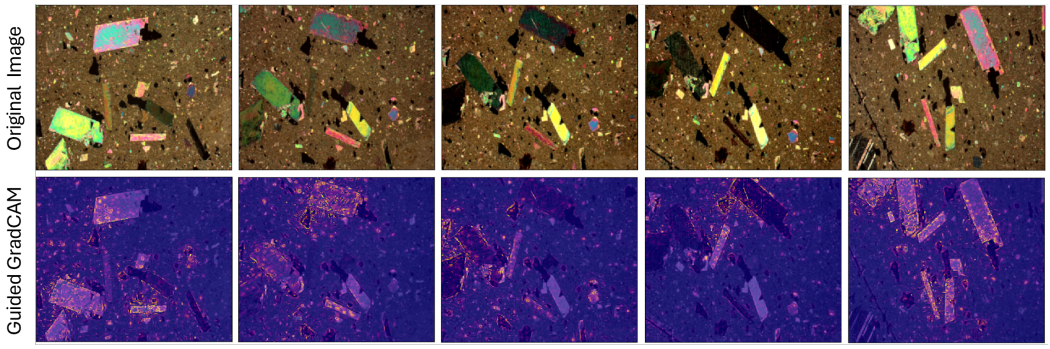


Figure 5.65. Guided Grad-CAM applied to the rotation sequence of sample TF4 from calcite *fabric*.

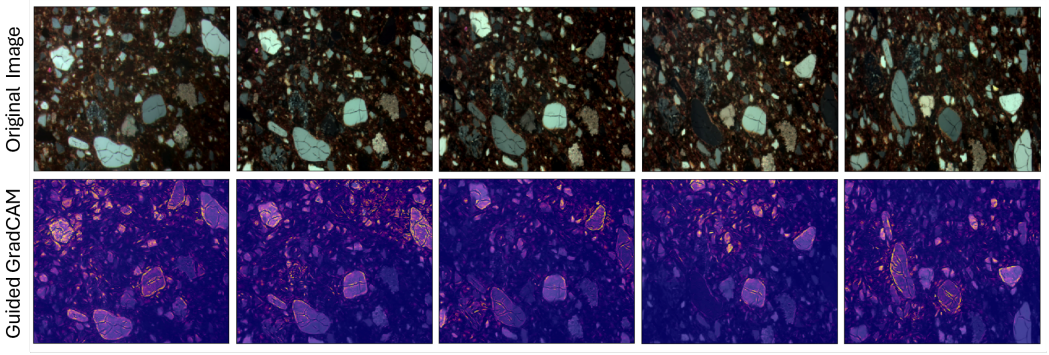


Figure 5.66. Guided Grad-CAM applied to the rotation sequence of sample TFMcamp1 from quartz *fabric*.

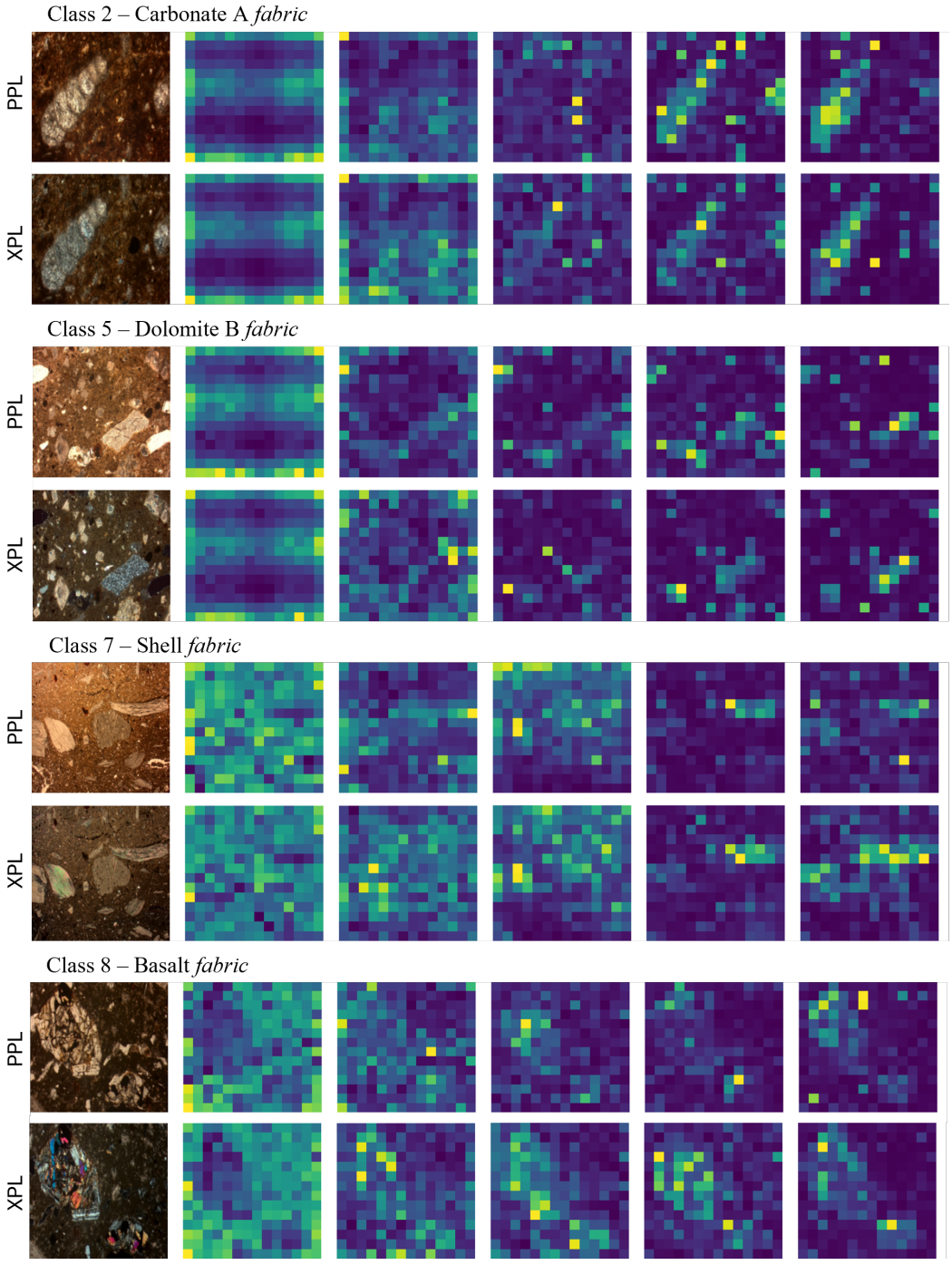


Figure 5.67. Attention maps of representative samples from carbonate A (sample KB146/30), dolomite B (sample K9), shell (sample KI11) and basalt (sample KB5) *fabrics* both in PPL and XPL. Visualization of head 3 for class 2 and 5, and head 5 for class 7 and 8, across ViT's layers.

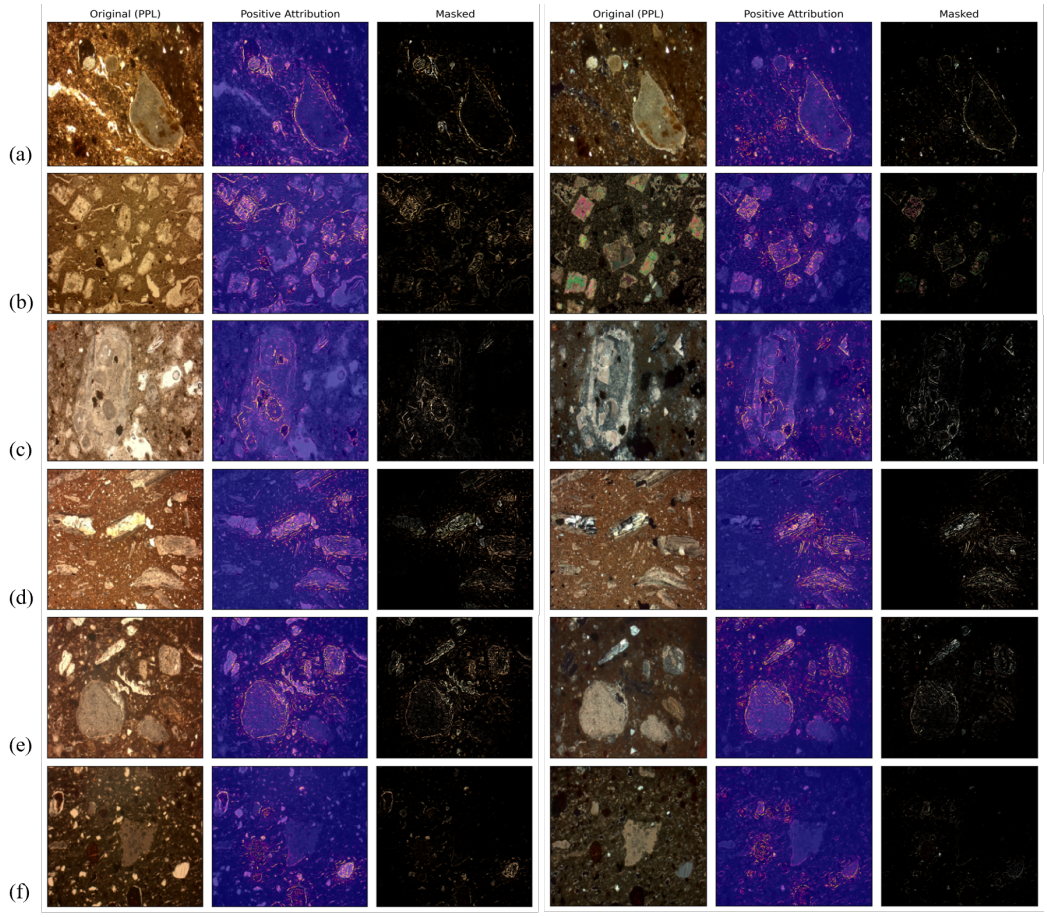


Figure B1. Guided Grad-CAM applied to pre-trained ResNet18. The figure shows the original images, the heatmaps overlapped to the thin-section images, and the masked images highlighting only the areas considered important for classification. (a) Class 3 (sample 718/25): Carbonate B *fabric*; (b) Class 4 (sample B1/14): Dolomite A *fabric*; (c) Class 5 (sample Kcamp15): Dolomite B *fabric*; (d) Class 7 (sample KIcamp2): Shell *fabric*; (e) Class 9 (sample K1): Carbonate C *fabric*; (f) Classe 10 (sample KB383/38): Carbonate D *fabric*.

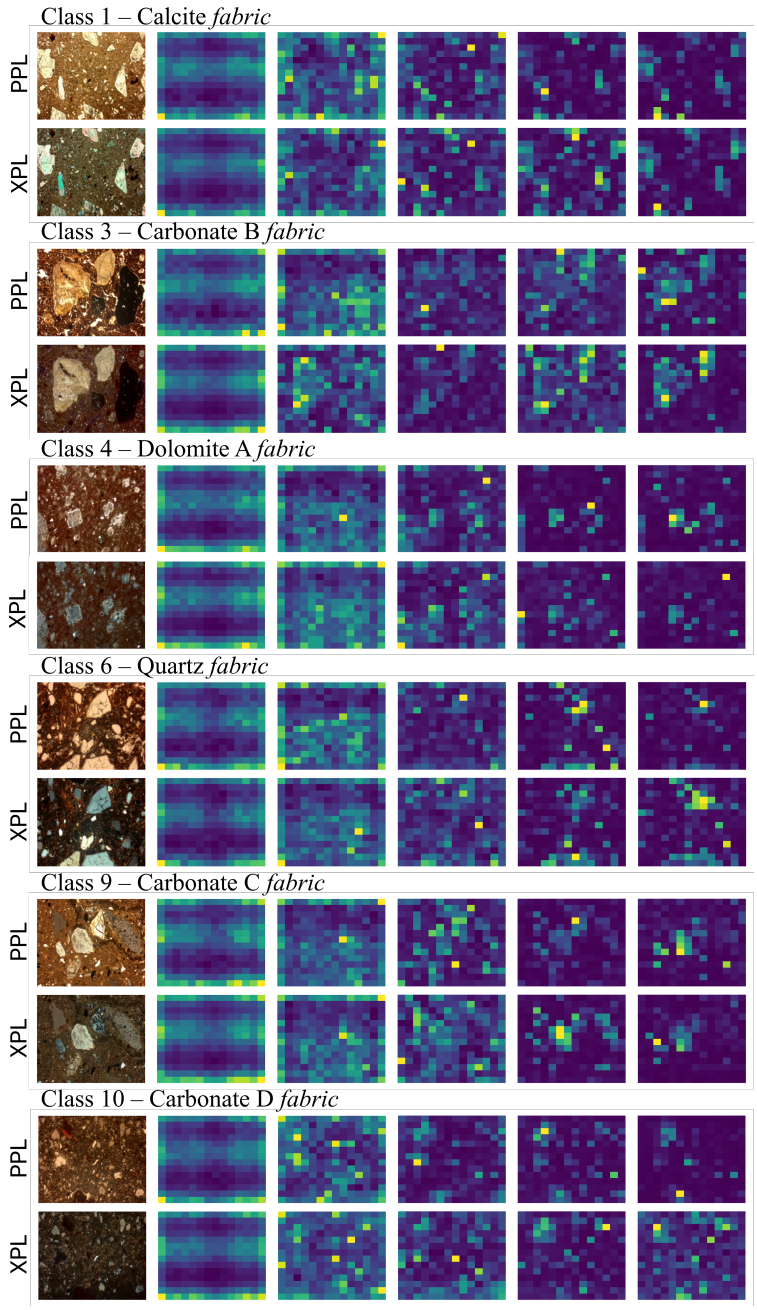


Figure B2. Attention maps of representative samples from calcite (sample TFcamp19), carbonate B (sample TFcamp13), dolomite A (sample Asur5), quartz (sample KBC4), carbonate C (sample K16) and carbonate D (sample KB314/113) *fabrics* both in PPL and XPL. Visualization of head 3 across the ViT's layers.

Chapter 6

Conclusion and Future Directions

This thesis has explored three fundamental and interconnected challenges in modern artificial intelligence: computational efficiency, adaptive inference, and model interpretability. Through a collection of works spanning multiple domains—from natural language processing and computer vision to high-energy physics and archaeological analysis—we have demonstrated that these challenges are not only addressable but can be tackled synergistically to create AI systems that are simultaneously more efficient, flexible, and transparent.

6.1 Summary of Contributions

The research presented in this thesis makes several key contributions to the field of artificial intelligence, organized around three core themes:

Adaptive Computation and Conditional Inference We established a comprehensive framework for conditional computation in neural networks [518], providing a unified formalism to describe techniques that dynamically activate or deactivate parts of their computational graph based on input characteristics. This theoretical foundation underpins much of the subsequent work in this thesis. Building on these principles, we introduced **Adaptive Computation Modules (ACMs)** [610], a generic architectural component that dynamically adjusts computational load on a per-token basis. By employing a sequence of progressive learners with learned gating mechanisms, ACMs achieve significant reductions in inference costs while maintaining competitive accuracy across computer vision and speech recognition tasks. This work demonstrates that granular conditional computation can be effectively integrated into existing architectures through distillation techniques. For the specific challenge of fine-tuning large vision transformers, we developed **ALaST (Adaptive Layer Selection Fine-Tuning)** [155], which adaptively estimates layer importance during training and allocates compute budgets accordingly. By selectively freezing layers or reducing token counts, ALaST achieves speedup in training time, reduction in FLOPs, and decrease in memory usage compared to full fine-tuning, while remaining compatible with parameter-efficient methods like LoRA. Finally, we used conditional computation in the context of next generation 6G networks [156] for efficient inference and transmission.

Efficient Inference for Large Language Models The deployment of large language models is fundamentally constrained by the memory requirements of the Key-Value (KV) cache, which grows linearly with sequence length. We addressed this bottleneck through complementary approaches. First, we discovered that the L_2 norm of key embeddings exhibits a strong correlation with attention scores, leading to a simple yet effective compression strategy [153]. This method achieves 50% KV cache reduction on language modeling tasks and up to 90% on passkey retrieval without accuracy loss, while maintaining compatibility with FlashAttention. Building on these insights, we developed **Q-Filters** [208], which exploits geometric properties of Query and Key vectors to enable training-free KV cache compression through context-agnostic projections. Q-Filters achieves 99% accuracy on needle-in-a-haystack tasks with 32× compression and reduces generation perplexity drop by up to 65% compared to Streaming-LLM, while remaining compatible with modern attention implementations. Finally, we introduced Expected Attention [151], a state-of-art method for KV Cache compression, that uses the distribution of future queries to compress the KV Cache.

Interpretable and Transparent AI Systems Understanding and controlling the internal mechanisms of neural networks is crucial for building trustworthy AI systems. We made several contributions to model interpretability across different architectures and application domains. For large language models, we investigated how they handle knowledge conflicts between parametric memory and contextual information [680]. By analyzing the residual stream, we demonstrated that LLMs internally register conflict signals at mid-layers, and these patterns differ systematically based on which knowledge source the model relies upon. Building on this analysis, we developed **SpARE** [678], a training-free representation engineering method that uses sparse auto-encoders to steer knowledge selection behavior at inference time, outperforming existing methods by up to 15%. We also uncovered the phenomenon of **attention sinks in diffusion language models** [506], revealing that while these models exhibit attention concentration similar to autoregressive models, their sinks display dynamic positional shifts during generation and show greater robustness to sink removal. These findings highlight fundamental differences in attention allocation between diffusion-based and autoregressive architectures.

In application domains, we demonstrated the value of interpretable architectures through two case studies. For high-energy particle physics, we developed a **Mixture-of-Experts Graph Transformer** [200] that achieves competitive classification accuracy while providing interpretable outputs through attention maps and expert specialization that align with known physics principles. For archaeological analysis, we applied explainable CNNs and Vision Transformers to classify Levantine ceramic thin sections [87], showing that XAI techniques like Guided Grad-CAM successfully identify mineralogical features that align with expert petrographic analysis.

6.2 Broader Impact and Implications

The contributions of this thesis extend beyond individual technical innovations to address fundamental challenges facing the AI community:

Democratizing AI Through Efficiency By reducing the computational and memory requirements of large models, our work on adaptive computation and KV cache compression helps democratize access to state-of-the-art AI capabilities. These efficiency gains enable deployment on resource-constrained devices and reduce energy consumption. This is particularly important for edge computing scenarios, where real-time inference must occur on devices with strict power and memory constraints.

Bridging AI and Domain Sciences Our applications in archaeology, particle physics, and industrial monitoring demonstrate that modern AI techniques can be successfully adapted to domain-specific challenges when combined with appropriate interpretability mechanisms. The ability to explain model decisions in terms of domain-relevant features—mineralogical composition in ceramics, physics-informed patterns in particle collisions—builds trust and enables AI to serve as a genuine collaborative tool for domain experts rather than an opaque black box.

Towards Trustworthy and Controllable AI The interpretability methods developed in this thesis, particularly in [678] through representation engineering, represent steps toward AI systems that are not only transparent but also controllable. The ability to detect and resolve knowledge conflicts, understand attention mechanisms, and intervene in model decisions at inference time without retraining provides practitioners with practical tools for deploying AI systems responsibly.

Unifying Efficiency and Interpretability A recurring theme throughout this thesis is that efficiency and interpretability need not be competing objectives. Adaptive computation mechanisms often inherently provide interpretability through their selection decisions—which tokens, layers, or experts are activated reveals what the model considers important. Similarly, interpretability analyses can guide efficiency improvements by identifying redundant computations or less critical components. This synergy suggests that future AI systems should be designed with both objectives in mind from the outset.

6.3 Limitations and Open Challenges

While this thesis makes significant progress on multiple fronts, several limitations and open challenges remain:

Generalization Across Architectures and Domains Many of our methods are tailored to specific architectures (e.g., transformers) or modalities (e.g., vision, language). While we have demonstrated cross-domain applicability in several cases, more work is needed to develop truly architecture-agnostic and modality-agnostic approaches to adaptive computation and interpretability. The principles may generalize, but their implementations often require careful adaptation.

Trade-offs and Multi-Objective Optimization Our work often focuses on optimizing specific trade-offs (e.g., accuracy vs. efficiency, or parametric vs. contextual knowledge). However, real-world deployments involve multiple competing objectives simultaneously:

accuracy, latency, memory, energy consumption, interpretability, fairness, and robustness. Developing principled multi-objective optimization frameworks that can navigate these complex trade-off spaces remains an important challenge.

Scalability of Interpretability Methods As models continue to grow in size and complexity, many interpretability methods face scalability challenges. Analyzing residual streams, computing attention maps, or training sparse auto-encoders for models with hundreds of billions of parameters presents computational and methodological difficulties. Developing interpretability techniques that scale gracefully with model size is crucial for understanding and controlling future AI systems.

6.4 Future Research Directions

Building on the contributions and limitations identified above, we propose several promising directions for future research:

Toward a general theory of conditional computation While we have provided a formalism for conditional computation, developing a more comprehensive theoretical framework that encompasses different granularities of adaptation (token, layer, module, network) and provides guarantees on efficiency-accuracy trade-offs would be valuable. Such a theory could guide architecture design and help predict when adaptive methods will be most beneficial.

Interpretability-driven architecture search Rather than applying interpretability methods post-hoc, future work could integrate interpretability objectives directly into neural architecture search or meta-learning procedures. This could lead to models that are interpretable by design, where the architecture itself reflects semantic structure in the task domain.

Multimodal adaptive systems As AI systems increasingly process multiple modalities simultaneously (vision, language, audio, sensory data), developing adaptive methods that can dynamically allocate computation across modalities based on task requirements and input characteristics presents an interesting challenge. Which modality deserves more computational resources may vary by input and over time during processing.

Continual and lifelong learning with adaptive computation Adaptive methods could play a crucial role in continual learning scenarios by dynamically allocating capacity to new tasks while preserving performance on previous tasks.

6.5 Concluding Remarks

The rapid advancement of artificial intelligence has brought both tremendous opportunities and significant challenges. As models grow larger and are deployed in increasingly critical applications—from scientific research to healthcare to autonomous systems—the need for efficiency, adaptability, and interpretability becomes paramount.

This thesis has demonstrated that these objectives are not only achievable but deeply interconnected. Adaptive computation mechanisms make models more efficient while providing interpretable signals about their decision-making process. Interpretability methods reveal opportunities for efficiency improvements by identifying redundant or less critical computations. Domain-specific applications show that these general principles can be successfully specialized to meet the unique requirements of diverse fields.

The path forward requires continued innovation across multiple fronts: developing more sophisticated theoretical frameworks, creating more powerful and scalable methods, and demonstrating impact in real-world applications. It also requires interdisciplinary collaboration, bringing together expertise in machine learning and domain sciences to create AI systems that are not only powerful but also understandable, controllable, and aligned with human needs.

As we stand at the frontier of increasingly capable AI systems, the work presented in this thesis contributes to a vision of artificial intelligence that is efficient enough to be widely accessible and interpretable enough to be trusted and understood. While significant challenges remain, the progress documented here provides a foundation for building the next generation of AI systems.

Bibliography

[1] Samira Abnar, Omid Saremi, Laurent Dinh, Shantel Wilson, Miguel Angel Bautista, Chen Huang, Vimal Thilak, Etai Littwin, Jiatao Gu, Josh Susskind, et al. Adaptivity and modularity for efficient generalization over task complexity. *arXiv preprint arXiv:2310.08866*, 2023.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[4] J. Adebayo, J. Gilmer, M. Muelly, I. J. Goodfellow andM. Hardt, and B. Kim. Sanity checks for saliency maps. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 9525–9536, 2018. https://doi.org/10.48550/arXiv.1810.03292.

[5] Garvita Agarwal, Lauren Hay, Ia Iashvili, Benjamin Mannix, Christine McLean, Margaret Morris, Salvatore Rappoccio, and Ulrich Schubert. Explainable ai for ml jet taggers using expert variables and layerwise relevance propagation. *Journal of High Energy Physics*, 2021, 2021. https://doi.org/10.1088/1742-6596/2438/1/012082.

[6] Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. Knowledge distillation from internal representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7350–7357, 2020.

[7] Kareem Ahmed, Zhe Zeng, Mathias Niepert, and Guy Van den Broeck. SIMPLE: A gradient estimator for k-subset sampling. *arXiv preprint arXiv:2210.01941*, 2022.

[8] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[9] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[10] Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontañón, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, et al. Colt5: Faster long-range transformers with conditional computation. *arXiv preprint arXiv:2303.09752*, 2023.

[11] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 1, context-free grammar. *arXiv preprint arXiv:2305.13673*, 2023.

[12] Muhammad Alrabeiah, Yu Zhang, and Ahmed Alkhateeb. Neural networks based beam codebooks: Learning mmwave massive mimo beams that adapt to deployment and hardware. *IEEE Transactions on Communications*, 70(6):3818–3833, 2022.

[13] Mohammad Mohammadi Amiri and Deniz Gündüz. Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air. *IEEE Transactions on Signal Processing*, 68:2155–2169, 2020.

[14] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–48, 2016.

[15] A. Anglisano, L. Casas, I. Queralt, and R. Di Febo. Supervised Machine Learning Algorithms to Predict Provenance of Archaeological Pottery Fragments. *Sustainability*, 14, 2022.

[16] Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[17] Alan Ansell, Edoardo Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1778–1796, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[18] Anthropic. System card: Claude opus 4 & claude sonnet 4. *arxiv*, 2025.

[19] AI Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 2024.

[20] A. Aprile, G. Castellano, and G. Eramo. Combining image analysis and modular neural networks for classification of mineral inclusions and pores in archaeological potsherds. *Journal of Archaeological Science*, 50:262–272, 2014.

[21] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M. Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. *arXiv preprint at arXiv::1912.06670*, 2020.

[22] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. 2016.

[23] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil Lopez, D. Molina, R. Benjamins, R. Chaila, and F. Herrera. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[24] Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[25] ATLAS Collaboration. Object-based missing transverse momentum significance in the ATLAS detector. In *10th International Workshop on Boosted Object Phenomenology, Reconstruction and Searches in HEP*.

[26] ATLAS Collaboration. The atlas experiment at the cern large hadron collider. *Journal of Instrumentation*, 3(8), 2008. https://doi.org/10.1088/1748-0221/3/08/S08003.

[27] ATLAS Collaboration. The LHCb Detector at the LHC. *Journal of Instrumentation*, 3(8), 2008. https://doi.org/10.1088/1748-0221/3/08/S08005.

[28] ATLAS Collaboration. Search for direct production of electroweakinos in final states with one lepton, jets and missing transverse momentum in pp collisions at $\sqrt{s} = 13$ tev with the atlas detector. *The European Physical Journal C*, 2020. https://doi.org/10.1140/epjc/s10052-020-8050-3.

[29] ATLAS Collaboration. Search for direct production of electroweakinos in final states with one lepton, jets and missing transverse momentum in pp collisions at $\sqrt{s} = 13$ tev with the atlas detector. *Journal of High Energy Physics*, 2023, 2023. https://doi.org/10.1007/JHEP12(2023)167.

[30] ATLAS Collaboration. Software performance of the atlas track reconstruction for lhc run 3. *Computing and Software for Big Science*, 8, 2024. https://doi.org/10.1007/s41781-023-00111-y.

[31] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

[32] Arun Babu, Changhan Wang, Andros Tjandra, Kushal Lakhotia, Qiantong Xu, Naman Goyal, Kritika Singh, Patrick von Platen, Yatharth Saraf, Juan Pino, Alexei Baevski, Alexis Conneau, and Michael Auli. Xls-r: Self-supervised cross-lingual speech representation learning at scale. *arXiv preprint at arXiv::2111.09296*, 2021.

[33] K. Badreshany, G. Philip, and M. Kennedy. The development of integrated regional economies in the Early Bronze Age Levant: new evidence from 'Combed Ware' jars. *Levant*, 42(1-2):160–196, 2020.

[34] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint at arXiv::2006.11477*, 2020.

[35] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. 2023.

[36] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

[37] Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. Improving the accuracy of early exits in multi-exit architectures via curriculum learning. In *Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[38] P. Ballirano, C. De Vito, L. Medeghini, S. Mignardi, V. Ferrini, P. Matthiae, D. Bersani, and P. Lottici. A combined use of optical microscopy, X-ray powder diffraction and micro-Raman spectroscopy for the characterization of ancient ceramic from Ebla (Syria). *Ceramics International*, 40(10):16409–16419, 2014.

[39] Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, February 2025.

[40] Federico Barbero, Andrea Banino, Steven Kapturowski, Dharshan Kumaran, João Guilherme Madeira Araújo, Alex Vitvitskyi, Razvan Pascanu, and Petar Veličković. Transformers need glasses! information over-squashing in language tasks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[41] Federico Barbero, Álvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Michael Bronstein, Petar Veličković, and Razvan Pascanu. Why do llms attend to the first token?, 2025.

[42] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[43] Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022. https://doi.org/10.1162/coli_a_00422.

[44] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[45] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[46] Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. 2015.

[47] Yoshua Bengio. Deep learning of representations: Looking forward. In *International conference on statistical language and speech processing*, pages 1–37. Springer, 2013.

[48] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

[49] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. 2013.

[50] Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for AI safety - A review. *CoRR*, abs/2404.14082, 2024.

[51] Baolong Bi, Shenghua Liu, Yiwei Wang, Lingrui Mei, and Xueqi Cheng. Is factuality enhancement a free lunch for llms? better factuality can lead to worse context-faithfulness. *arXiv preprint arXiv:2404.00216*, 2024.

[52] Fadi Biadsy, Youzheng Chen, Xia Zhang, Oleg Rybakov, Andrew Rosenberg, and Pedro Moreno. A scalable model specialization framework for training and inference using submodels and its application to speech model personalization. In *Interspeech 2022*. ISCA, 2022.

[53] Chenghong Bian, Yulin Shao, Haotian Wu, Emre Ozfatura, and Deniz Gündüz. Process-and-forward: Deep joint source-channel coding over cooperative relay networks. *IEEE Journal on Selected Areas in Communications*, 2025.

[54] S. H. Bickler. Machine Learning Arrives in Archaeology. *Advances in Archaeological Practice*, 9(2):186–191, 2021.

[55] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

[56] Ahsan Bilal, David Ebert, and Beiyu Lin. Llms for explainable AI: A comprehensive survey. *CoRR*, abs/2504.00125, 2025.

[57] Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. 2023. *OpenAI Blog*, 14, 2023.

[58] Francesco Binucci, Paolo Banelli, Paolo Di Lorenzo, and Sergio Barbarossa. Multi-user goal-oriented communications with energy-efficient edge resource management. *IEEE Transactions on Green Communications and Networking*, 7(4):1709–1724, 2023.

[59] Francesco Binucci, Mattia Merluzzi, Paolo Banelli, Emilio Calvanese Strinati, and Paolo Di Lorenzo. Enabling edge artificial intelligence via goal-oriented deep neural network splitting. In *2024 19th International Symposium on Wireless Communication Systems (ISWCS)*, pages 1–6, 2024.

[60] I. Bird. Computing for the large hadron collider. *Annual Review of Nuclear and Particle Science*, 61:99–118, 2011. https://doi.org/10.1146/annurev-nucl-102010-130059.

[61] Daniel Biś, Maksim Podkorytov, and Xiuwen Liu. Too much in common: Shifting of embeddings in transformer language models and its implications. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5117–5130, Online, June 2021. Association for Computational Linguistics.

[62] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[63] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *NIPS*, pages 4349–4357, 2016.

[64] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 527–536. JMLR.org, 2017.

[65] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

[66] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. 2023.

[67] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. 2023.

[68] Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4598–4602, 2023.

[69] Andrea Bontempelli, Stefano Teso, Katya Tentori, Fausto Giunchiglia, and Andrea Passerini. Concept-level debugging of part-prototype networks. *arXiv preprint arXiv:2205.15769*, 2022.

[70] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. 2014.

[71] M. Botticelli, S. Mignardi, C. De Vito, Y. W. Liao, D. Montanari, M. Shakarna, L. Nigro, and L. Medeghini. Variability in pottery production at Khalet al-Jam'a necropolis, Bethlehem (West Bank): From the Early-Middle Bronze to the Iron Age. *Ceramics International*, 46(10):16405–16415, 2020.

[72] M. Botticelli, S. Mignardi, C. De Vito, M. Sala, G. Mazzotta, L. Da Silva Gondim, and L. Medeghini. The 'metallic ware' from Tell el Far'ah North (West Bank): Petrography, technology, and provenance of a hidden ceramic industry. *Ceramics International*, 48(1):1366–1374, 2022.

[73] Eirina Bourtsoulatze, David Burth Kurka, and Deniz Gündüz. Deep joint source-channel coding for wireless image transmission. *IEEE Trans. on Cognitive Communications and Netw.*, 5(3):567–579, 2019.

[74] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L. Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

[75] Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[76] Han Cai, Chuang Gan, and Song Han. Efficientvit: Enhanced linear attention for high-resolution low-computation visual recognition. 2022.

[77] Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. Tinytl: Reduce memory, not parameters for efficient on-device learning. 33:11285–11297, 2020.

[78] Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model. In *Forty-first International Conference on Machine Learning*, 2024.

[79] Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling, 2024.

[80] Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv*, 2025.

[81] Zhihang Cai, Xingjun Zhang, Zhendong Tan, and Zheng Wei. Nqkv: A kv cache quantization scheme based on normal distribution characteristics. *arXiV*, 2025.

[82] Emilio Calvanese Strinati and Sergio Barbarossa. 6g networks: Beyond shannon towards semantic and goal-oriented communications. *Computer Networks*, 190:107930, 2021.

[83] Emilio Calvanese Strinati, Paolo Di Lorenzo, Vincenzo Sciancalepore, Adnan Aijaz, Marios Kountouris, Deniz Gündüz, Petar Popovski, Mohamed Sana, Photios A. Stavrou, and et al. Goal-oriented and semantic communication in 6g ai-native networks: The 6g-goals approach. 2024.

[84] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

[85] Nicola Cancedda. Spectral filters, dark signals, and attention sinks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4792–4808, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[86] Nicola Cancedda. Spectral filters, dark signals, and attention sinks, 2024.

[87] S Capriotti, A Devoto, S Scardapane, S Mignardi, and L Medeghini. Interpretable classification of levantine ceramic thin sections via neural networks. *Machine Learning: Science and Technology*, 2025.

[88] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, pages 9630–9640. IEEE, 2021.

[89] G. Caspari and P. Crespo. Convolutional neural networks for archaeological site detection–finding "princely" tombs. *Journal of Archaeological Science*, 110, 2019.

[90] D. Castelvecchi. Can we open the black box of AI? *Nature*, 538(7623):20–23, 2016.

[91] Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. Improving steering vectors by targeting sparse autoencoder features. *ArXiv*, abs/2411.02193, 2024.

[92] Beitao Chen, Xinyu Lyu, Lianli Gao, Jingkuan Song, and Heng Tao Shen. Alleviating hallucinations in large vision-language models through hallucination-induced optimization. *CoRR*, abs/2405.15356, 2024.

[93] Canyu Chen and Kai Shu. Can llm-generated misinformation be detected? *arXiv preprint arXiv:2309.13788*, 2023.

[94] Canyu Chen and Kai Shu. Combating misinformation in the age of llms: Opportunities and challenges. *AI Magazine*, 2023.

[95] Hung-Ting Chen, Michael J. Q. Zhang, and Eunsol Choi. Rich knowledge sources bring complex knowledge conflicts: Recalibrating models to reflect conflicting evidence. In *EMNLP*, pages 2292–2307. Association for Computational Linguistics, 2022.

[96] Jiakang Chen, Selim F. Yilmaz, Di You, Pier Luigi Dragotti, and Deniz Gündüz. SING: Semantic Image Communications using Null-Space and INN-Guided Diffusion Models. 2025.

[97] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint*, 2021.

[98] Quan Chen, Song Guo, Kaijia Wang, Wenchao Xu, Jing Li, Zhipeng Cai, Hong Gao, and Albert Y. Zomaya. Towards real-time inference offloading with distributed edge computing: The framework and algorithms. *IEEE Transactions on Mobile Computing*, 23(7):7552–7571, 2024.

[99] Shiqi Chen, Miao Xiong, Junteng Liu, Zhengxuan Wu, Teng Xiao, Siyang Gao, and Junxian He. In-context sharpness as alerts: An inner representation perspective for hallucination mitigation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[100] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. 2022.

[101] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016. https://doi.org/10.1145/2939672.2939785.

[102] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 19974–19988, 2021.

[103] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. 2016.

[104] Xuanyao Chen, Zhijian Liu, Haotian Tang, Li Yi, Hang Zhao, and Song Han. Sparsevit: Revisiting activation sparsity for efficient high-resolution vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2061–2070, June 2023.

[105] Yida Chen, Fernanda B. Viégas, and Martin Wattenberg. Beyond surface statistics: Scene representations in a latent diffusion model. *CoRR*, abs/2306.05720, 2023.

[106] Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023.

[107] Zhourong Chen, Yang Li, Samy Bengio, and Si Si. You look twice: Gaternet for dynamic filter selection in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9172–9180, 2019.

[108] M. S. Chesson and G. Philip. Tales of the City? "Urbanism" in the Early Bronze Age Levant from Mediterranean and Levantine Perspectives. *Horizon*, 1(2):26–37, 2003.

[109] Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, et al. On the representation collapse of sparse mixture of experts. *Advances in Neural Information Processing Systems*, 35:34600–34613, 2022.

[110] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[111] Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[112] Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In *ICLR*. OpenReview.net, 2024.

[113] Bilal Chughtai, Alan Cooney, and Neel Nanda. Summing up the facts: Additive mechanisms behind factual recall in llms. *CoRR*, abs/2402.07321, 2024.

[114] Aidan Clark, Diego De Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, George Bm Van Den Driessche, Eliza Rutherford, Tom Hennigan, Matthew J Johnson, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Marc'Aurelio Ranzato, Jack Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. Unified scaling laws for routed language models. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 4057–4086. PMLR, 17–23 Jul 2022.

[115] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[116] Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*, 2018.

[117] J.-B. Cordonnier, A. Loukas, and M. Jaggi. On the relationship between self-attention and convolutional layers. In *8th International Conference on Learning Representations*, 2020. https://doi.org/10.48550/arXiv.1911.03584.

[118] Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. Adaptively sparse transformers. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184. Association for Computational Linguistics, November 2019.

[119] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*, 2014.

[120] P. Crochet and P. Braun-Munzinger. Investigation of background subtraction techniques for high mass dilepton physics. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 484:564–572, 2002. https://doi.org/10.1016/S0168-9002(01)02005-8.

[121] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020.

[122] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *CoRR*, abs/2309.08600, 2023.

[123] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

[124] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *ACL (1)*, pages 8493–8502. Association for Computational Linguistics, 2022.

[125] Jincheng Dai, Wang Sixian, Kailin Tan, Zhongwei Si, Xiaoqi Qin, Kai Niu, and Ping Zhang. Nonlinear transform source-channel coding for semantic communications. pages 1–1, 08 2022.

[126] Jincheng Dai, Sixian Wang, Kailin Tan, Zhongwei Si, Xiaoqi Qin, Kai Niu, and Ping Zhang. Nonlinear transform source-channel coding for semantic communications. *IEEE Journal on Selected Areas in Communications*, 40(8):2300–2316, 2022.

[127] M. D'Andrea. *The Southern Levant in Early Bronze IV: issues and perspectives in the pottery evidence.* Sapienza Univ. di Roma, Dipt. di Scienze dell'Antichità 20XX, 2015.

[128] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[129] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. *International Conference on Learning Representations (ICLR)*, 2024.

[130] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[131] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[132] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *Proceedings of the 2024 International Conference on Learning Representations (ICLR)*, 2024.

[133] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations*, 2024.

[134] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations*, 2024.

[135] Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 4599–4610. Association for Computational Linguistics, 2021.

[136] Patil Deepak and Elmeleegy Amr. How to scale your model. https://cloud.google.com/blog/products/compute/ai-inference-recipe-using-nvidia-dynamo-with-ai-hypercomputer, 2024.

[137] Amr Elmeleegy Deepak Patil. Fast and efficient ai inference with new nvidia dynamo recipe on ai hypercomputer. https://jax-ml.github.io/scaling-book/, 2024.

[138] DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv*, 2024.

[139] DeepSeek-AI. Deepseek-v3 technical report, 2024.

[140] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[141] DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024.

[142] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme Ruiz, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd Van Steenkiste, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Collier, Alexey A. Gritsenko, Vighnesh Birodkar, Cristina Nader Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetic, Dustin Tran, Thomas Kipf, Mario Lucic, Xiaohua Zhai, Daniel Keysers, Jeremiah J. Harmsen, and Neil Houlsby. Scaling vision transformers to 22 billion parameters. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7480–7512. PMLR, 23–29 Jul 2023.

[143] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *IEEE conference on computer vision and pattern recognition*, pages 248–255., 2009.

[144] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. pages 248–255, 2009.

[145] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE Computer Society, 2009.

[146] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. 2022.

[147] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

[148] Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pages 7750–7774. PMLR, 2023.

[149] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[150] A Devoto, F Alvetreti, J Pomponi, P Di Lorenzo, P Minervini, et al. Adaptive layer selection for efficient vision transformer fine-tuning. *Neurocomputing*, 654, 2025.

[151] A Devoto, M Jeblick, and S Jégou. Expected attention: Kv cache compression by estimating attention from future queries distribution. arXiv preprint arXiv:2510.00636, 2025.

[152] A Devoto, I Spinelli, F Murabito, C Chiovoloni, R Musmeci, and S Scardapane. Reidentification of objects from aerial photos with hybrid siamese neural networks. *IEEE Transactions on Industrial Informatics*, 19:2997–3005, 2025.

[153] A Devoto, Y Zhao, S Scardapane, and P Minervini. A simple and effective l_2 norm-based strategy for kv cache compression. In *Empirical Methods in Natural Language Processing (EMNLP 2024)*, 2024.

[154] Alessio Devoto, Federico Alvetreti, Jary Pomponi, Paolo Di Lorenzo, Pasquale Minervini, and Simone Scardapane. Adaptive layer selection for efficient vision transformer fine-tuning, 2024.

[155] Alessio Devoto, Federico Alvetreti, Jary Pomponi, Paolo Di Lorenzo, Pasquale Minervini, and Simone Scardapane. Adaptive layer selection for efficient vision transformer fine-tuning. 2024.

[156] Alessio Devoto, Simone Petruzzi, Jary Pomponi, Paolo Di Lorenzo, and Simone Scardapane. Adaptive semantic token selection for ai-native goal-oriented communications, 2024.

[157] Alessio "Devoto, Yu Zhao, Simone Scardapane, and Pasquale" Minervini. A simple and effective l_2 norm-based strategy for KV cache compression. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18476–18499, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[158] Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective l_2 norm-based strategy for kv cache compression. *The 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

[159] Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective l_2 norm-based strategy for KV cache compression. *CoRR*, abs/2406.11430, 2024.

[160] G. DeZoort, P. W. Battaglia, C. Biscarat, and J.-R. Vlimant. Graph neural networks at the large hadron collider. *Nature Reviews Physics*, 5(5):281–303, 2023. https://doi.org/10.1038/s42254-023-00569-0.

[161] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M. Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7865–7885. PMLR, 23–29 Jul 2023.

[162] Paolo Di Lorenzo, Mattia Merluzzi, Francesco Binucci, Claudio Battiloro, Paolo Banelli, Emilio Calvanese Strinati, and Sergio Barbarossa. Goal-oriented communications for the iot: System design and adaptive resource optimization. *IEEE Internet of Things Magazine*, 6(4):26–32, 2023.

[163] Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

[164] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. 2020.

[165] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, Zhai X., T. Unterthiner, M. Dehghani, M Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929*, 2020.

[166] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[167] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.

[168] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.

[169] Yibing Du, Antoine Bosselut, and Christopher D. Manning. Synthetic disinformation attacks on automated fact verification systems. In *AAAI*, pages 10581–10589. AAAI Press, 2022.

[170] J. Duarte and J.-R. Vlimant. *Graph Neural Networks for Particle Tracking and Reconstruction*, chapter 12, pages 387–436. World Scientific, 2020. https://doi.org/10.1142/9789811234033_0012.

[171] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

[172] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. In *AAAI'21 Workshop on Deep Learning on Graphs: Methods and Applications*, 2021. https://doi.org/10.48550/arXiv.2012.09699.

[173] D. Eigen, M. A. Ranzato, and I. Sutskever. Learning factored representations in a deep mixture of experts. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014. https://doi.org/10.48550/arXiv.1312.4314.

[174] H. El-Hajj, O. Eberle, A. Merklein, A. Siebold, N. Shlomi, J. Büttner, J. Martinetz, K. R. Müller, G. Montavon, and M. Valleriani. Explainability and transparency in the realm of digital humanities: toward a historian XAI. *International Journal of Digital Humanities*, 5(2):299–331, 2023.

[175] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

[176] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

[177] Shohei Enomoro and Takeharu Eda. Learning to cascade: Confidence calibration for improving the accuracy and computational cost of cascade inference systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7331–7339, 2021.

[178] S Ercolino, A Devoto, L Monorchio, M Santini, S Mazzaro, and S Scardapane. On the robustness of vision transformers for in-flight monocular depth estimation. *Industrial Artificial Intelligence*, 1, 2025.

[179] N. Farsad, M. Rao, and A. Goldsmith. Deep learning for joint source-channel coding of text. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2326–2330, 2018.

[180] Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Jürgen Gall. Adaptive token sampling for efficient vision transformers. In *European Conference on Computer Vision*, pages 396–414, 2022.

[181] William Fedus, Jeff Dean, and Barret Zoph. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*, 2022.

[182] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

[183] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23(1), 2022.

[184] Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *arXiv*, 2024.

[185] Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. Identify critical KV cache in LLM inference from an output perturbation perspective. 2025.

[186] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1039–1048, 2017.

[187] Quentin Fournier, Gaétan Marceau Caron, and Daniel Aloise. A practical survey on faster and lighter transformers. *ACM Computing Surveys*, 55(14s):1–40, 2023.

[188] Yao Fu. Challenges in deploying long-context transformers: A theoretical peak performance analysis. *arXiv preprint arXiv:2405.08944*, 2024.

[189] Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128k context. *arXiv preprint arXiv:2402.10171*, 2024.

[190] Huiguo Gao, Guanding Yu, and Yunlong Cai. Adaptive modulation and retransmission scheme for semantic communication systems. *IEEE Transactions on Cognitive Communications and Networking*, 2023.

[191] Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*, 2019.

[192] L. Gao, T. D. la Tour, H. Tillman, G. Goh, R. Troll, A. Radford, I. Sutskever, J. Leike, and J. Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

[193] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[194] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *CoRR*, abs/2406.04093, 2024.

[195] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024.

[196] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@NeurIPS 2023)*, 2023.

[197] AP Gema, JOJ Leang, G Hong, A Devoto, ACM Mancino, R Saxena, X He, et al. Are we done with mmlu? In *NAACL 2025 - Nations of the Americas Chapter of the Association for...*, 2025.

[198] GeminiTeam. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv*, 2025.

[199] GemmaTeam. Gemma 3. *ArXiV*, 2025.

[200] D Genovese, A Sgroi, A Devoto, S Valentine, L Wood, C Sebastiani, et al. Mixture-of-experts graph transformers for interpretable particle collision detection. *Nature Scientific Reports*, 2025.

[201] D. Genovese, A. Sgroi, A. Devoto, S. Valentine, L. Wood, C. Sebastiani, S. Giagu, M. D'Onofrio, and S. Scardapane. Mixture-of-Experts Graph Transformers for Interpretable Particle Collision Detection., 2025.

[202] Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 12216–12235. Association for Computational Linguistics, 2023.

[203] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, 2021.

[204] Amir Ghodrati, Babak Ehteshami Bejnordi, and Amirhossein Habibian. Frameexit: Conditional early exiting for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15608–15618, 2021.

[205] A. Ghorbani, A. Abid, and J. Y. Zou. Interpretation of neural networks is fragile. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 3681–3688, 2019. https://doi.org/10.1609/aaai.v33i01.33013681.

[206] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89, 2018.

[207] Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7b ssm hybrid model. *arXiv*, 2024.

[208] N Godey, A Devoto, Y Zhao, S Scardapane, P Minervini, E de la Clergerie, et al. Q-filters: Leveraging qk geometry for efficient kv cache compression. In *SLLM @ ICLR 2025*, 2025.

[209] Nathan Godey, Éric Clergerie, and Benoît Sagot. Anisotropy is inherent to self-attention in transformers. In Yvette Graham and Matthew Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 35–48, St. Julian's, Malta, March 2024. Association for Computational Linguistics.

[210] Nathan Godey, Alessio Devoto, Yu Zhao, Simone Scardapane, Pasquale Minervini, Éric de la Clergerie, and Benoît Sagot. Q-filters: Leveraging qk geometry for efficient kv cache compression. *arXiv*, 2025.

[211] Oded Goldreich, Brendan Juba, and Madhu Sudan. A theory of goal-oriented communication. *Association for Computing Machinery*, 59(2), 2012.

[212] Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. Scaling diffusion language models via adaptation from autoregressive models. In *The Thirteenth International Conference on Learning Representations*, 2023.

[213] Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models, 2025.

[214] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2022.

[215] Aleksa Gordić. Inside vllm: Anatomy of a high-throughput llm inference system. https://www.aleksagordic.com/blog/vllmAleksaGordiÄĞ, 2025.

[216] Y. Goren. The Southern Levant in the Early Bronze Age IV: The Petrographic Perspective. *Bulletin of the American Schools of Oriental Research*, 303(1):33–72, 1996.

[217] Y. Goren. Ceramic Technology and Provenance at Khirbat Iskandar. In S. Richard, J. C. Long, P. S. Holdorf, and G. Peterman, editors, *Archaeological Expedition to Khirbat Iskandar and its Environs, Jordan: Khirbat Iskandar Final Report on the Early Bronze IV Area C 'Gateway' and Cemeteries*, pages 133–140. The American Schools of Oriental Research, 2010.

[218] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

[219] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

[220] R. Greenberg. *Traveling in (world) time: transformation, commoditization, and the beginnings of urbanism in the Southern Levant.* Oxbow Books, 2011.

[221] R. Greenberg. *The archaeology of the bronze age levant.* Cambridge University Press, 2019.

[222] R. Greenberg and N. Porat. A Third Millennium Levantine Pottery Production Center: Typology, Petrography, and Provenance of the Metallic Ware of Northern Israel and Adjacent Regions. *Bulletin of the American Schools of Oriental Research*, 301(1):5–24, 1996.

[223] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers. 2024.

[224] Audrūnas Gruslys, Rémi Munos, Ivo Danihelka, Marc Lanctot, and Alex Graves. Memory-efficient backpropagation through time. page 4132–4140, 2016.

[225] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *2312.00752*, 2024.

[226] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *International Conference on Learning Represenations*, 2022.

[227] Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. *arXiv preprint arXiv:2410.10781*, 2024.

[228] Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. In *The Thirteenth International Conference on Learning Representations*, 2025.

[229] D. Guest, K. Cranmer, and D. Whiteson. Deep learning and its application to lhc physics. *Annual Review of Nuclear and Particle Science*, 68:161–181, 2018. https://doi.org/10.1146/annurev-nucl-101917-021019.

[230] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5), August 2018.

[231] Gündüz, Deniz and Wigger, Michèle A and Tung, Tze-Yang and Zhang, Ping and Xiao, Yong. Joint source–channel coding: Fundamentals and recent progress in practical designs. *Proceedings of the IEEE*, 2024.

[232] Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. Attention score is not all you need for token importance indicator in kv cache reduction: Value also matters, 2024.

[233] Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. Attention score is not all you need for token importance indicator in KV cache reduction: Value also matters. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21158–21166, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[234] Yoav Gur-Arieh, Roy Mayan, Chen Agassy, Atticus Geiger, and Mor Geva. Enhancing automated interpretability with output-centric feature descriptions. *arXiv preprint arXiv:2501.08319*, 2025.

[235] Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. DEMix layers: Disentangling domains for modular language modeling. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages

5557–5576, Seattle, United States, July 2022. Association for Computational Linguistics.

[236] A. Guyot, M. Lennon, T. Lorho, and L. Hubert-Moy. Combined Detection and Segmentation of Archeological Structures from LiDAR Data Using a Deep Learning Approach. *Journal of Computer Applications in Archaeology*, 4(1):1, 2021.

[237] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. 2016.

[238] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[239] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456, 2021.

[240] Yizeng Han, Yifan Pu, Zihang Lai, Chaofei Wang, Shiji Song, Junfeng Cao, Wenhui Huang, Chao Deng, and Gao Huang. Learning to weight samples for dynamic early-exiting networks. In *European Conference on Computer Vision*, pages 362–378. Springer, 2022.

[241] Vikas Hassija, Vinay Chamola, Atmesh Mahapatra, Abhinandan Singal, Divyansh Goel, Kaizhu Huang, Simone Scardapane, Indro Spinelli, Mufti Mahmud, and Amir Hussain. Interpreting black-box models: a review on explainable artificial intelligence. *Cognitive Computation*, 16(1):45–74, 2024.

[242] Joakim Bruslund Haurum, Sergio Escalera, Graham W Taylor, and Thomas B Moeslund. Which tokens to use? investigating token reduction in vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 773–783, 2023.

[243] Jakob Drachmann Havtorn, Amélie Royer, Tijmen Blankevoort, and Babak Ehteshami Bejnordi. Msvit: Dynamic mixed-scale tokenization for vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 838–848, October 2023.

[244] Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed Chi. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *Advances in Neural Information Processing Systems*, 34:29335–29347, 2021.

[245] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778., 2016.

[246] Xuanli He, Iman Keivanloo, Yi Xu, Xiang He, Belinda Zeng, Santosh Rajagopalan, and Trishul Chilimbi. Magic pyramid: Accelerating inference with early exiting and token pruning. In *NeurIPS 2021 Workshop on Efficient Natural Language and Speech Processing*, 2021.

[247] Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[248] Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 9318–9333. Association for Computational Linguistics, 2023.

[249] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[250] Charles Herrmann, Richard Strong Bowen, and Ramin Zabih. Channel selection using Gumbel Softmax. In *European Conference on Computer Vision*, pages 241–257. Springer, 2020.

[251] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[252] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[253] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Núria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics, 2020.

[254] Connor Holmes, Masahiro Tanaka, Michael Wyatt, Ammar Ahmad Awan, Jeff Rasley, Samyam Rajbhandari, Reza Yazdani Aminabadi, Heyang Qin, Arash Bakhtiari, Lev Kurilenko, et al. Deepspeed-fastgen: High-throughput text generation for llms via mii and deepspeed-inference. *arXiv preprint arXiv:2401.08671*, 2024.

[255] Giwon Hong, Jeonghwan Kim, Junmo Kang, Sung-Hyon Myaeng, and Joyce Jiyoung Whang. Why so gullible? enhancing the robustness of retrieval-augmented models against counterfactual noise. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 2474–2495. Association for Computational Linguistics, 2024.

[256] Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts llm performance, 2025.

[257] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465, 2021.

[258] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 2024.

[259] Le Hou, Richard Yuanzhe Pang, Tianyi Zhou, Yuexin Wu, Xinying Song, Xiaodan Song, and Denny Zhou. Token dropping for efficient BERT pretraining. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3774–3784, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[260] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. 97:2790–2799, 09–15 Jun 2019.

[261] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.

[262] Jeremy Howard. Imagenette.

[263] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

[264] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

[265] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[266] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. 2022.

[267] Huaibo Huang, Xiaoqiang Zhou, Jie Cao, Ran He, and Tieniu Tan. Vision transformer with super token sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22690–22699, 2023.

[268] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *CoRR*, abs/2311.05232, 2023.

[269] Wenhui Huang, Yanxin Zhou, Xiangkun He, and Chen Lv. Goal-guided transformer-enabled reinforcement learning for efficient autonomous navigation. *IEEE Transactions on Intelligent Transportation Systems*, 25(2):1832–1845, 2023.

[270] Yufei Huang, Shengding Hu, Xu Han, Zhiyuan Liu, and Maosong Sun. Unified view of grokking, double descent and emergent abilities: A perspective from circuits competition. *arXiv preprint arXiv:2402.15175*, 2024.

[271] Robert Huben, Hoagy Cunningham, Logan Riggs, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *ICLR*. OpenReview.net, 2024.

[272] Iris AM Huijben, Wouter Kool, Max B Paulus, and Ruud JG Van Sloun. A review of the Gumbel-max trick and its extensions for discrete stochasticity in machine learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1353–1371, 2022.

[273] C. Hörr, E. Lindinger, and Brunnett G. Machine learning based typology development in archaeology. *Journal on Computing and Cultural Heritage*, 7(1):1–23, 2014.

[274] Shadi Iskander, Kira Radinsky, and Yonatan Belinkov. Shielded representations: Protecting sensitive attributes through iterative gradient-based projection. In *ACL (Findings)*, pages 5961–5977. Association for Computational Linguistics, 2023.

[275] Gagan Jain, Nidhi Hegde, Aditya Kusupati, Arsha Nagrani, Shyamal Buch, Prateek Jain, Anurag Arnab, and Sujoy Paul. Mixture of nested experts: Adaptive processing of visual tokens. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[276] Gagan Jain, Nidhi Hegde, Aditya Kusupati, Arsha Nagrani, Shyamal Buch, Prateek Jain, Anurag Arnab, and Sujoy Paul. Mixture of nested experts: Adaptive processing of visual tokens. 2024.

[277] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2017.

[278] Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems*, 34:9895–9907, 2021.

[279] Simon Jegou and Maximilian Jeblick. Kvpress, 2024.

[280] Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. *International Conference on Machine Learning*, 2024.

[281] Frederick Jelinek. Interpolated estimation of markov source parameters from sparse data. In *Proc. Workshop on Pattern Recognition in Practice, 1980*, 1980.

[282] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[283] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[284] Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[285] Yeong-Hwa Jin, Keon-Ho Lee, and Dong-Wan Choi. Querynet: Querying neural networks for lightweight specialized models. *Information Sciences*, 589:186–198, 2022.

[286] Zhuoran Jin, Pengfei Cao, Hongbang Yuan, Yubo Chen, Jiexin Xu, Huaijun Li, Xiaojian Jiang, Kang Liu, and Jun Zhao. Cutting off the head ends the conflict: A mechanism for interpreting and mitigating knowledge conflicts in language models. *arXiv preprint arXiv:2402.18154*, 2024.

[287] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the em algorithm. 1993.

[288] X. Ju, S. Farrell, P. Calafiura, D. Murnane, M. Prabhat, L. Gray, T. Klijnsma, G. Cerati K. Pedro, J. Kowalkowski, G. Perdue, P. Spentzouris, N. Tran, J.-R. Vlimant, A. Zlokapa, J. Pata, M. Spiropulu, S. An, A. Aurisano, and T. Usher. Graph neural networks for particle reconstruction in high energy physics detectors. In *33rd Conference on Neural Information Processing Systems, Workshop on Machine Learning and the Physical Sciences*, 2019. https://doi.org/10.48550/arXiv.2003.11603.

[289] Brendan Juba and Madhu Sudan. Universal semantic communication i. In *Proceedings of the fortieth annual ACM symposium on theory of computing*, pages 123–132, 2008.

[290] Greg Kamradt. Needle in a haystack - pressure testing llms. https://github.com/gkamradt/LLMTest_NeedleInAHaystack, 2023.

[291] Greg Kamradt. Needle in a haystack - pressure testing llms. https://github.com/gkamradt/LLMTest_NeedleInAHaystack, 2023.

[292] Nikhil Kandpal, Brian Lester, Mohammed Muqeeth, Anisha Mascarenhas, Monty Evans, Vishal Baskaran, Tenghao Huang, Haokun Liu, and Colin Raffel. Git-theta: a git extension for collaborative development of machine learning models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[293] Gal Kaplun, Andrey Gurevich, Tal Swisa, Mazor David, Shai Shalev-Shwartz, and Eran Malach. Subtuning: Efficient finetuning for multi-task learning. 2023.

[294] Polina Karpikova, Ekaterina Radionova, Anastasia Yaschenko, Andrei Spiridonov, Leonid Kostyushko, Riccardo Fabbricatore, and Aleksei Ivakhnenko. Fiancee: Faster inference of adversarial networks via conditional early exits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12032–12043, 2023.

[295] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics, 2020.

[296] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *NeurIPS*, pages 852–863, 2021.

[297] R. Kashefi, L. Barekatain, M. Sabokrou, and F. Aghaeipoor. Explainability of Vision Transformers: A Comprehensive Review and New Perspectives. *arXiv preprint arXiv:2311.06786*, 2023.

[298] Jaeyeon Kim, Lee Cheuk-Kit, Carles Domingo-Enrich, Yilun Du, Sham Kakade, Timothy Ngotiaoco, Sitan Chen, and Michael Albergo. Any-order flexible length masked diffusion. *arXiv preprint arXiv:2509.01025*, 2025.

[299] Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. Kvzip: Query-agnostic kv cache compression with context reconstruction, 2025.

[300] Minchul Kim, Shangqian Gao, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. Token fusion: Bridging the gap between token pruning and token merging. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1383–1392, January 2024.

[301] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization., 2014.

[302] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

[303] Diederik P Kingma. Adam: A method for stochastic optimization. 2014.

[304] Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Trans. Assoc. Comput. Linguistics*, 6:317–328, 2018.

[305] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, Minghai Qin, and Yanzhi Wang. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 620–640, Cham, 2022. Springer Nature Switzerland.

[306] Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*, pages 3499–3508. PMLR, 2019.

[307] Wouter Kool, Herke Van Hoof, and Max Welling. Ancestral gumbel-top-k sampling for sampling without replacement. *The Journal of Machine Learning Research*, 21(1):1726–1761, 2020.

[308] Alexandros Kouris, Stylianos I Venieris, Stefanos Laskaridis, and Nicholas Lane. Multi-exit semantic segmentation networks. In *European Conference on Computer Vision*, pages 330–349. Springer, 2022.

[309] Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. BERT busters: Outlier dimensions that disrupt transformers. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3392–3405, Online, August 2021. Association for Computational Linguistics.

[310] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). 2009.

[311] Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. Booksum: A collection of datasets for long-form narrative summarization. *ACL*, 2022.

[312] BartŁ Krzepkowski, Monika Michaluk, Franciszek Szarwacki, Piotr Kubaty, Jary Pomponi, Bartosz WĂŁjcik, Kamil Adamczewski, et al. Joint or disjoint: Mixing training regimes for early-exit models. 2024.

[313] Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, Prateek Jain, et al. Matformer: Nested transformer for elastic inference. *arXiv preprint arXiv:2310.07707*, 2023.

[314] D. B. Kurka and D. Gündüz. DeepJSCC-f: Deep joint source-channel coding of images with feedback. *IEEE Journal on Selected Areas in Information Theory*, 1(1):178–193, 2020.

[315] D. B. Kurka and D. Gündüz. Bandwidth-agile image transmission with deep joint source-channel coding. *IEEE Transactions on Wireless Communications*, 20(12):8081–8095, 2021.

[316] David Burth Kurka and Deniz Gündüz. Bandwidth-agile image transmission with deep joint source-channel coding. *IEEE Transactions on Wireless Communications*, 20(12):8081–8095, 2021.

[317] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022.

[318] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

[319] Ibtissam Labriji, Mattia Merluzzi, Fatima Ezzahra Airod, and Emilio Calvanese Strinati. Energy-efficient cooperative inference via adaptive deep neural network splitting at the edge. In *ICC 2023 - IEEE International Conference on Communications*, pages 1712–1717, 2023.

[320] Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. Mercury: Ultra-fast language models based on diffusion. *arXiv*, 2025.

[321] Michael Lan, Philip Torr, Austin Meek, Ashkan Khakzar, David Krueger, and Fazl Barez. Sparse autoencoders reveal universal feature spaces across large language models. *arXiv preprint arXiv:2410.06981*, 2024.

[322] Sander Land and Max Bartolo. Fishing for magikarp: Automatically detecting under-trained tokens in large language models, 2024.

[323] S. Lapuschkin, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10, 2015. https://doi.org/10.1371/journal.pone.0130140.

[324] Stefanos Laskaridis, Stylianos I. Venieris, Mario Almeida, Ilias Leontiadis, and Nicholas D. Lane. Spinn: synergistic progressive inference of neural networks over device and cloud. 2020.

[325] Luzian Lebovitz, Lukas Cavigelli, Michele Magno, and Lorenz K Muller. Efficient inference with model cascades. *Transactions on Machine Learning Research*, 2023.

[326] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521(7553):436–444, 2015.

[327] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[328] Jaeduk Lee, Hojung Lee, and Wan Choi. Wireless channel adaptive dnn split inference for resource-constrained edge devices. *IEEE Communications Letters*, 27(6):1520–1524, 2023.

[329] Yoonho Lee, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. *International Conference on Learning Representations*, 2023.

[330] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen. Gshard: Scaling giant models with conditional computation and

automatic sharding. In *Proceedings of the 9th International Conference on Learning Representations*, 2021. https://doi.org/10.48550/arXiv.2006.16668.

[331] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. In *Proceedings of the 2021 International Conference on Learning Representations (ICLR)*, 2021.

[332] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. pages 3045–3059, November 2021.

[333] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[334] Bo Li, Yifei Shen, Jingkang Yang, Yezhen Wang, Jiawei Ren, Tong Che, Jun Zhang, and Ziwei Liu. Sparse mixture-of-experts are domain generalizable learners. *arXiv preprint arXiv:2206.04046*, 2022.

[335] Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic slimmable network. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 8607–8617, 2021.

[336] Dongfang Li, Zhenyu Liu, Xinshuo Hu, Zetian Sun, Baotian Hu, and Min Zhang. In-context learning state vector with inner and momentum optimization. *CoRR*, abs/2404.11225, 2024.

[337] Haoyang LI, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole HU, Wei Dong, Li Qing, and Lei Chen. A survey on large language model acceleration based on KV cache management. *Transactions on Machine Learning Research*, 2025.

[338] Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. Beyond fixed: Training-free variable-length denoising for diffusion large language models. *arXiv preprint arXiv:2508.00819*, 2025.

[339] Kenneth Li, Oam Patel, Fernanda B. Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *NeurIPS*, 2023.

[340] Kenneth Li, Oam Patel, Fernanda B. Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[341] Tianyi Li, Mingda Chen, Bowei Guo, and Zhiqiang Shen. A survey on diffusion language models, 2025.

[342] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343, 2022.

[343] Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In *ACL (1)*, pages 12286–12312. Association for Computational Linguistics, 2023.

[344] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. pages 4582–4597, 2021.

[345] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. pages 4582–4597, August 2021.

[346] Youwei Li, Kai Zhang, Jie Cao, Radu Timofte, and Luc Van Gool. Not all patches are what you need: Expediting vision transformers via token reorganizations. 2022.

[347] Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir H. Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and Lili Qiu. SCBench: A KV cache-centric analysis of long-context methods. In *The Thirteenth International Conference on Learning Representations*, 2025.

[348] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[349] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 2025.

[350] Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning, 2023.

[351] Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. Stack more layers differently: High-rank training through low-rank updates. *arXiv preprint arXiv:2307.05695*, 2023.

[352] Youwei Liang, Chongjian GE, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. EVit: Expediting vision transformers via token reorganizations. 2022.

[353] Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca D. Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *CoRR*, abs/2408.05147, 2024.

[354] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[355] Y. LiHua, Z. WenBo, and Q. WangRen. Chronological classification of ming and qing dynasty ceramics images based on an enhanced resnet50 model. *STAR: Science & Technology of Archaeological Research*, 11(1):e2498260, 2025.

[356] D. Lim, J. D. Robinson, L. Zhao, T. E. Smidt, S. Sra, H. Maron, and S. Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *The 11th International Conference on Learning Representations*, 2023. https://doi.org/10.48550/arXiv.2202.13013.

[357] Bin Lin, Tao Peng, Chen Zhang, Minmin Sun, Lanbo Li, Hanyu Zhao, Wencong Xiao, Qi Xu, Xiafei Qiu, Shen Li, et al. Infinite-llm: Efficient llm service for long context with distattention and distributed kvcache. *arXiv preprint arXiv:2401.02669*, 2024.

[358] Haokun Lin, Haobo Xu, Yichen Wu, Ziyu Guo, Renrui Zhang, Zhichao Lu, Ying Wei, Qingfu Zhang, and Zhenan Sun. Quantization meets dllms: A systematic study of post-training quantization for diffusion llms. *arXiv preprint arXiv:2508.14896*, 2025.

[359] Min Lin, Jie Fu, and Yoshua Bengio. Conditional computation for continual learning. *arXiv preprint arXiv:1906.06635*, 2019.

[360] Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. Fq-vit: Post-training quantization for fully quantized vision transformer. pages 1173–1179, 2022.

[361] Zehui Lin, Liwei Wu, Mingxuan Wang, and Lei Li. Learning language specific sub-network for multilingual machine translation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 293–305, Online, August 2021. Association for Computational Linguistics.

[362] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. 2019.

[363] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. 2023.

[364] James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. Training-free activation sparsity in large language models, 2025.

[365] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 11:157–173, 2024.

[366] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 2024.

[367] Sheng Liu, Haotian Ye, Lei Xing, and James Y. Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. In *ICML*. OpenReview.net, 2024.

[368] Tianlin Liu, Mathieu Blondel, Carlos Riquelme, and Joan Puigcerver. Routers in vision mixture of experts: An empirical study. *arXiv preprint arXiv:2401.15969*, 2024.

[369] Xiaoran Liu, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. Longllada: Unlocking long context capabilities in diffusion llms, 2025.

[370] Yifei Liu, Mathias Gehrig, Nico Messikommer, Marco Cannici, and Davide Scaramuzza. Revisiting token pruning for object detection and instance segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2658–2668, 2024.

[371] Wing Fei Lo, Nitish Mital, Haotian Wu, and Deniz Gündüz. Collaborative semantic communication for edge inference. *IEEE Wireless Communications Letters*, 12(7):1125–1129, 2023.

[372] Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7052–7063, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[373] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4765–4774, 2017. https://doi.org/10.48550/arXiv.1705.07874.

[374] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang. Parameterized explainer for graph neural network. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020. https://doi.org/10.48550/arXiv.2011.04573.

[375] Shi Luohe, Hongyi Zhang, Yao Yao, Zuchao Li, and hai zhao. Keep the cost down: A review on methods to optimize LLM's KV-cache consumption. In *First Conference on Language Modeling*, 2024.

[376] M. Lyons. Ceramic Fabric Classification of Petrographic Thin Sections with Deep Learning. *Journal of computer applications in archaeology*, 4(1):188–201, 2021.

[377] M. Lyons, F. Fecher, and M. Reindel. From LiDAR to deep learning: A case study of computer-assisted approaches to the archaeology of Guadalupe and northeast Honduras. *it - Information Technology*, 64(6):233–246, 2022.

[378] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[379] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 9802–9822. Association for Computational Linguistics, 2023.

[380] Yuyi Mao et al. Green edge ai: A contemporary survey. *Proceedings of the IEEE*, 112(7):880–911, 2024.

[381] Dmitrii Marin, Jen-Hao Rick Chang, Anurag Ranjan, Anish Prabhu, Mohammad Rastegari, and Oncel Tuzel. Token pooling in vision transformers. 2021.

[382] L. Maritan, C. Mazzoli, V. Michielin, D. Morandi Bonacossi, M. Luciani, and G. Molin. The provenance and production technology of Bronze Age and Iron Age pottery from Tell Mishrifeh/Qatna (Syria). *Archaeometry*, 47(4):723–744, 2005.

[383] Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

[384] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Computing Surveys*, 55(5):1–30, 2022.

[385] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Comput. Surv.*, 55(5), 2022.

[386] Alexey Gritsenko et al. Matthias Minderer. Simple open-vocabulary object detection with vision transformers. 2022.

[387] Thomas McGrath, Andrei Kapishnikov, Nenad Tomasev, Adam Pearce, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in alphazero. *CoRR*, abs/2111.09259, 2021.

[388] L. Medeghini, L. Fabrizi, S. De Vito, C. Mignardi, L. Nigro, and C. Gallo, E. Fiaccavento. The ceramic of the "Palace of the Copper Axes" (Khirbet al-Batrawy, Jordan): A palatial special production. *Ceramics International*, 42(5):5952–5962, 2016.

[389] L. Medeghini, S. Mignardi, C. De Vito, N. Macro, M. D'Andrea, and S. Richard. New insights on Early Bronze Age IV pottery production and consumption in the southern Levant: The case of Khirbat Iskandar, Jordan. *Ceramics International*, 42(16):18991–19005, 2016.

[390] L. Medeghini and L. Nigro. Khirbet al-Batrawy ceramics: a systematic mineralogical and petrographic study for investigating the material culture. *Periodico di Mineralogia*, 86(1), 2017.

[391] L. Medeghini, M. Sala, C. De Vito, and S. Mignardi. A forgotten centre of ceramic production in Southern Levant: Preliminary analytical study of the Early Bronze Age pottery from Tell el-Far'ah North (West Bank). *Ceramics International*, 45(9):11457–11467, 2019.

[392] Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. 2022.

[393] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[394] Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. Adavit: Adaptive vision transformers for efficient image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12309–12318, 2022.

[395] Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. Adavit: Adaptive vision transformers for efficient image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12309–12318, 2022.

[396] Mattia Merluzzi, Paolo Di Lorenzo, and Sergio Barbarossa. Wireless edge machine learning: Resource allocation and trade-offs. *IEEE Access*, 9:45377–45398, 2021.

[397] William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. *International Conference on Machine Learning*, 2024.

[398] MetaAI. Introducing llama 4: Advancing multimodal intelligence. *arXiv*, 2024.

[399] MetaAI. The llama 3 herd of models. *arXiv*, 2025.

[400] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: a survey. *Transactions on Machine Learning Research*, 6:1–35, 2023.

[401] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. 2018.

[402] Tomás Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751. The Association for Computational Linguistics, 2013.

[403] Evan Miller. Attention is off by one. https://www.evanmiller.org/attention-is-offby-one.html, 2023.

[404] Julian Minder, Kevin Du, Niklas Stoehr, Giovanni Monea, Chris Wendler, Robert West, and Ryan Cotterell. Controllable context sensitivity and the knob behind it. *arXiv preprint arXiv:2411.07404*, 2024.

[405] MiniMax, Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, Enwei Jiao, Gengxin Li, Guojun Zhang, Haohai Sun, Houze Dong, Jiadai Zhu, Jiaqi Zhuang, Jiayuan Song, Jin Zhu, Jingtao Han, Jingyang Li, Junbin Xie, Junhao Xu, Junjie Yan, Kaishun Zhang, Kecheng Xiao, Kexi Kang, Le Han, Leyang Wang, Lianfei Yu, Liheng Feng, Lin Zheng, Linbo Chai, Long Xing, Meizhi Ju, Mingyuan Chi, Mozhi Zhang, Peikai Huang, Pengcheng Niu, Pengfei Li, Pengyu Zhao, Qi Yang, Qidi Xu, Qiexiang Wang, Qin Wang, Qiuhui Li, Ruitao Leng, Shengmin Shi, Shuqi Yu, Sichen Li, Songquan Zhu, Tao Huang, Tianrun Liang, Weigao Sun, Weixuan Sun, Weiyu Cheng, Wenkai Li, Xiangjun Song, Xiao Su, Xiaodong Han, Xinjie Zhang, Xinzhu Hou, Xu Min, Xun Zou, Xuyang Shen, Yan Gong, Yingjie Zhu, Yipeng Zhou, Yiran Zhong, Yongyi Hu, Yuanxiang Fan, Yue Yu, Yufeng Yang, Yuhao Li, Yunan Huang, Yunji Li, Yunpeng Huang, Yunzhi Xu, Yuxin Mao, Zehan Li, Zekang Li, Zewei Tao, Zewen Ying, Zhaoyang Cong, Zhen Qin, Zhenhua Fan, Zhihang Yu, Zhuo Jiang, and Zijia Wu. Minimax-01: Scaling foundation models with lightning attention, 2025.

[406] Sarthak Mittal, Yoshua Bengio, and Guillaume Lajoie. Is a modular architecture enough? *Advances in Neural Information Processing Systems*, 35:28747–28760, 2022.

[407] Muqeeth Mohammed, Haokun Liu, and Colin Raffel. Models with conditional computation learn suboptimal solutions. In *I Can't Believe It's Not Better Workshop: Understanding Deep Learning Through Empirical Falsification*, 2022.

[408] Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

[409] Farouk Mokhtar, Raghav Kansal, Daniel Diaz, Javier Duarte, Joosep Pata, Maurizio Pierini, and Jean-Roch Vlimant. Explaining machine-learned particle-flow reconstruction. In *35th Conference on Neural Information Processing Systems*, 2021. https://doi.org/10.48550/arXiv.2111.12840.

[410] Christoph Molnar. *Interpretable Machine Learning*. Leanpub, 3 edition, 2025.

[411] G. Montavon, W. Samek, and K. R. Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[412] Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *arXiv*, 2025.

[413] Timur Mudarisov, Mikhail Burtsev, Tatiana Petrova, and Radu State. Limitations of normalization in attention mechanism. *arXiv:2508.17821*, 2025.

[414] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

[415] Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. Learning to route among specialized experts for zero-shot generalization. *arXiv preprint arXiv:2402.05859*, 2024.

[416] Mohammed Muqeeth, Haokun Liu, and Colin Raffel. Soft merging of experts with adaptive routing. *arXiv preprint arXiv:2306.03745*, 2023.

[417] MG Sarwar Murshed, Christopher Murphy, Daqing Hou, Nazar Khan, Ganesh Ananthanarayanan, and Faraz Hussain. Machine learning at the network edge: A survey. *ACM Computing Surveys (CSUR)*, 54(8):1–37, 2021.

[418] B. Mustafa, C. Riquelme, J. Puigcerver, R. Jenatton, and N. Houlsby. Multimodal contrastive learning with limoe: the language-image mixture of experts. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022. https://doi.org/10.48550/arXiv.2206.02770.

[419] Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. Multimodal contrastive learning with limoe: the language-image mixture of experts. *Advances in Neural Information Processing Systems*, 35:9564–9576, 2022.

[420] P. Navarro, C. Cintas, M. Lucena, J. M. Fuertes, C. Delrieux, and M. Molinos. Learning feature representation of Iberian ceramics with automatic classification models. *Journal of Cultural Heritage*, 48:65–73, 2021.

[421] Piotr Nawrot, Adrian Lańcucki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In *Forty-first International Conference on Machine Learning*, 2024.

[422] Piotr Nawrot, Robert Li, Renjie Huang, Sebastian Ruder, Kelly Marchisio, and Edoardo M. Ponti. The sparse frontier: Sparse attention trade-offs in transformer llms. *arXiv*, 2025.

[423] Piotr Nawrot, Adrian Lańcucki, Marcin Chochowski, David Tarjan, and Edoardo M. Ponti. Dynamic memory compression: retrofitting llms for accelerated inference. *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[424] Michael Neely. *Stochastic network optimization with application to communication and queueing systems*. Morgan & Claypool Publishers, 2010.

[425] Michael Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*, volume 3. 01 2010.

[426] Vlad Niculae, Caio F Corro, Nikita Nangia, Tsvetomila Mihaylova, and André FT Martins. Discrete latent structure in neural networks. *arXiv preprint arXiv:2301.07473*, 2023.

[427] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv*, 2025.

[428] Mathias Niepert, Pasquale Minervini, and Luca Franceschi. Implicit MLE: backpropagating through discrete exponential family distributions. *Advances in Neural Information Processing Systems*, 34:14567–14579, 2021.

[429] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. Dec 2008.

[430] E. Nobile, M. Troiano, F. Mangini, M. Mastrogiuseppe, J. Verdi, F. Frezza, C. Conati Barbaro, and A. Gopher. Neural network analysis for predicting metrics of fragmented laminar artifacts: a case study from MPPNB sites in the Southern Levant. *Scientific Reports*, 14, 2024.

[431] Nostalgebraist. Interpreting gpt: the logit lens, 2023.

[432] NVIDIA. TensorRT-LLM. https://github.com/NVIDIA/TensorRT-LLM, 2024.

[433] Chris Olah. Distributed representations: Composition & superposition. *Transformer Circuits Thread*, 2023.

[434] Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious, 2025.

[435] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

[436] OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander

Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b and gpt-oss-20b model card, 2025.

[437] OpenAI. Learning to reason with large language models. https://openai.com/index/learning-to-reason-with-llms/, 2024.

[438] OpenAI. Introducing deep research. https://openai.com/index/introducing-deep-research/, 2025.

[439] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Trans. Mach. Learn. Res.*, 2024, 2024.

[440] Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state rnns. *arXiv*, 2024.

[441] C. Orton and M. Hughes. *Pottery in Archaeology*. Cambridge University Press, 2013.

[442] Francesco Ortu, Zhijing Jin, Diego Doimo, Mrinmaya Sachan, Alberto Cazzaniga, and Bernhard Schölkopf. Competition of mechanisms: Tracing how language models handle facts and counterfactuals. *arXiv preprint arXiv:2402.11655*, 2024.

[443] F. Oviedo, J. L. Ferres, T. Buonassisi, and K. T. Butler. Interpretable and Explainable Machine Learning for Materials Science and Chemistry. *Accounts of Materials Research*, 3(6):597–607, 2022.

[444] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red^2: Interpretability-aware redundancy reduction for vision transformers. *Advances in Neural Information Processing Systems*, 34:24898–24911, 2021.

[445] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red^2: Interpretability-aware redundancy reduction for vision transformers. 34:24898–24909, 2021.

[446] Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 8298–8319. Association for Computational Linguistics, 2023.

[447] Liangming Pan, Wenhu Chen, Min-Yen Kan, and William Yang Wang. Attacking open-domain question answering by injecting misinformation. In *IJCNLP (1)*, pages 525–539. Association for Computational Linguistics, 2023.

[448] Zizheng Pan, Jianfei Cai, and Bohan Zhuang. Stitchable neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16102–16112, 2023.

[449] Zizheng Pan, Bohan Zhuang, Haoyu He, Jing Liu, and Jianfei Cai. Less is more: Pay less attention in vision transformers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(2):2035–2043, Jun. 2022.

[450] Francesco Pappone. Attention sinks from the graph perspective. https://publish.obsidian.md/the-tensor-throne/Transformers+as+GNNs/Attention+sinks+from+the+graph+perspective, August 2025.

[451] Jihong Park, Sumudu Samarakoon, Mehdi Bennis, and Mérouane Debbah. Wireless network intelligence at the edge. *Proceedings of the IEEE*, 107(11):2204–2239, 2019.

[452] Junyoung Park, Dalton Jones, Matthew J Morse, Raghavv Goel, Mingu Lee, and Chris Lott. Keydiff: Key similarity-based kv cache eviction for long-context llm inference in resource-constrained environments. *arXiv*, 2025.

[453] N. Passalis, J. Raitoharju, A. Tefas, and M. Gabbouj. Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits. *Pattern Recognition*, 105:107346, 2020.

[454] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[455] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[456] Adam Paszke, Sam Gross, Francisco Massa, Gal Lerer, James Bradbury, Gregory Chillemi, Luca Antiga, Alban Desmaison, Andreas Tejani, Soumith Chilamkurthy, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv*, 2019.

[457] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. The carbon footprint of machine learning training will plateau, then shrink. *arXiv preprint arXiv:2204.05149*, 2022.

[458] L. M. Pawlowicz and C. E. Downum. Applications of deep learning to decorated ceramic typology and classification: A case study using Tusayan White Ware from Northeast Arizona. *Journal of Archaeological Science*, 130, 2021.

[459] PerplexityAI. Perplexity deep research. https://www.perplexity.ai/hub/blog/introducing-perplexity-deep-research, 2025.

[460] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. Language models as knowledge bases? In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2463–2473. Association for Computational Linguistics, 2019.

[461] R. Pezoa, L. Salinas, and C. Torres. Explainability of high energy physics events classification using shap. *Journal of Physics: Conference Series*, 2438, 2023. https://doi.org/10.1088/1742-6596/2438/1/012082.

[462] Jonas Pfeiffer, Gregor Geigle, Aishwarya Kamath, Jan-Martin Steitz, Stefan Roth, Ivan Vulić, and Iryna Gurevych. xGQA: Cross-lingual visual question answering. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2497–2511. Association for Computational Linguistics, May 2022.

[463] Jonas Pfeiffer, Naman Goyal, Xi Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. Lifting the curse of multilinguality by pre-training modular transformers. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3479–3495, Seattle, United States, July 2022.

[464] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503. Association for Computational Linguistics, April 2021.

[465] Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo Maria Ponti. Modular deep learning. *Transactions on Machine Learning Research*, 11:1–76, 2023.

[466] Mikołaj Piórczyński, Filip Szatkowski, Klaudia Bałazy, and Bartosz Wójcik. Exploiting transformer activation sparsity with dynamic inference. *arXiv preprint arXiv:2310.04361*, 2023.

[467] J Pomponi, A Devoto, and S Scardapane. Class incremental learning with probability dampening and cascaded gated classifier. *Neurocomputing*, 460, 2024.

[468] J Pomponi, M Merluzzi, A Devoto, MP Mota, P Di Lorenzo, and S Scardapane. Goal-oriented communications based on recursive early exit neural networks. In *Asilomar Conference on Signals, Systems, and Computers*, 2024.

[469] J. Pomponi, S. Scardapane, and A. Uncini. A probabilistic re-intepretation of confidence scores in multi-exit models. *Entropy*, 24:1, 2021.

[470] Jary Pomponi, Mattia Merluzzi, Alessio Devoto, Mateus Pontes Mota, Paolo Di Lorenzo, and Simone Scardapane. Goal-oriented communications based on recursive early exit neural networks, 2024.

[471] Edoardo M Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. Combining modular skills in multitask learning. *arXiv preprint arXiv:2202.13914*, 2022.

[472] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann. Explainability methods for graph convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10772–10781, 2019. https://doi.org/10.1109/CVPR.2019.01103.

[473] Ramya Prabhu, Ajay Nayak, Jayashree Mohan, Ramachandran Ramjee, and Ashish Panwar. vattention: Dynamic memory management for serving llms without pagedattention. *arXiv preprint arXiv:2405.04437*, 2024.

[474] G. Pruthi, F. Liu, S. Kale, and M. Sundararajan. Estimating training data influence by tracing gradient descent. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 19920–19930, 2020. https://doi.org/10.48550/arXiv.2002.08484.

[475] Raffaele Pugliese, Stefano Regondi, and Riccardo Marini. Machine learning-based approach: Global trends, research directions, and regulatory standpoints. *Data Science and Management*, 2021.

[476] Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. From sparse to soft mixtures of experts. *arXiv preprint arXiv:2308.00951*, 2023.

[477] Joan Puigcerver, Carlos Riquelme, Basil Mustafa, Cedric Renggli, André Susano Pinto, Sylvain Gelly, Daniel Keysers, and Neil Houlsby. Scalable transfer learning with expert models. In *Proceedings of the 2021 International Conference on Learning Representations (ICLR)*, 2021.

[478] Y. Qi, M. Z. Qiu, H. Z. Jing, Z. Q. Wang, C. L. Yu, J. F. Zhu, F. Wang, and T. Wang. End-to-end ancient ceramic classification toolkit based on deep learning: A case study of black glazed wares of Jian kilns (Song Dynasty, Fujian province). *Ceramics International*, 48(23):34516–34532, 2022.

[479] Li Qiao, Mahdi Boloursaz Mashhadi, Zhen Gao, and Deniz Gündüz. Token-domain multiple access: Exploiting semantic orthogonality for collision mitigation. 2025.

[480] Li Qiao, Mahdi Boloursaz Mashhadi, Zhen Gao, Rahim Tafazolli, Mehdi Bennis, and Dusit Niyato. Token communications: A unified framework for cross-modal context-aware semantic communications. 2025.

[481] Yifu Qiu, Zheng Zhao, Yftah Ziser, Anna Korhonen, Edoardo M. Ponti, and Shay B. Cohen. Spectral editing of activations for large language model alignment. *CoRR*, abs/2405.09719, 2024.

[482] Zihan Qiu, Zeyu Huang, and Jie Fu. Emergent mixture-of-experts: Can dense pre-trained transformers benefit from emergent modular structures? *arXiv preprint arXiv:2310.10908*, 2023.

[483] Enrique Queipo-de Llano, Álvaro Arroyo, Federico Barbero, Xiaowen Dong, Michael Bronstein, Yann LeCun, and Ravid Shwartz-Ziv. Attention sinks and compression valleys in llms are two sides of the same coin. *arXiv preprint arXiv:2510.06477*, 2025.

[484] P. S. Quinn. *Ceramic Petrography: The Interpretation of Archaeological Pottery & Related Artefacts in Thin Section*. Archaeopress Publishing, 2013.

[485] Quintanar, Álvaro and Izquierdo, Rubén and Parra, Ignacio and Fernández-Llorca, David. Goal-oriented transformer to predict context-aware trajectories in urban scenarios. *Engineering Proceedings*, 39(1), 2023.

[486] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[487] Alec Radford, Rafal Józefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *CoRR*, abs/1704.01444, 2017.

[488] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. 139:8748–8763, 18–24 Jul 2021.

[489] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy. Do Vision Transformers See Like Convolutional Neural Networks?. *Advances in neural information processing systems*, 34:12116–12128, 2021.

[490] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? 2021.

[491] Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *CoRR*, abs/2407.14435, 2024.

[492] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 18332–18346. PMLR, 2022.

[493] Janarthanan Rajendran, Aravind Lakshminarayanan, Mitesh M. Khapra, Prasanna P, and Balaraman Ravindran. Attend, adapt and transfer: Attentive deep architecture for adaptive transfer from multiple sources in the same domain. In *Proceedings of the 2017 International Conference on Learning Representations (ICLR)*, 2017.

[494] A. Ramil, A.J. López, and A. Yáñez. Application of artificial neural networks for the rapid classification of archaeological ceramics by means of laser induced breakdown spectroscopy (libs). *Applied Physics A*, pages 197–202, 2008.

[495] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

[496] Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In *ACL*, pages 7237–7256. Association for Computational Linguistics, 2020.

[497] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[498] Liliang Ren, Congcong Chen, Haoran Xu, Young Jin Kim, Adam Atkinson, Zheng Zhan, Jiankai Sun, Baolin Peng, Liyuan Liu, Shuohang Wang, Hao Cheng, Jianfeng Gao, Weizhu Chen, and Yelong Shen. Decoder-hybrid-decoder architecture for efficient reasoning with long generation. *arXiv*, 2025.

[499] Cedric Renggli, André Susano Pinto, Neil Houlsby, Basil Mustafa, Joan Puigcerver, and Carlos Riquelme. Learning to merge tokens in vision transformers. *arXiv preprint arXiv:2202.12015*, 2022.

[500] Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. *CoRR*, abs/2312.06681, 2023.

[501] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

[502] Morgane Rivière, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos,

Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozinska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucinska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju-yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjösund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, and Lilly McNealus. Gemma 2: Improving open language models at a practical size. *CoRR*, abs/2408.00118, 2024.

[503] Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34:17555–17566, 2021.

[504] Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. In *Proceedings of the 2018 International Conference on Learning Representations (ICLR)*, 2018.

[505] Brian C Ross. Mutual information between discrete and continuous data sets. *PloS one*, 9(2):e87357, 2014.

[506] Maximo Eduardo Rulli, Simone Petruzzi, Edoardo Michielon, Fabrizio Silvestri, Simone Scardapane, and Alessio Devoto. Attention sinks in diffusion language models, 2025.

[507] G. Ruschioni, D. Malchiodi, A. M. Zanaboni, and L. Bonizzoni. Supervised learning algorithms as a tool for archaeology: Classification of ceramic samples described by chemical element concentrations. *ournal of Archaeological Science*, 49, 2023.

[508] Valeria Ruscio, Umberto Nanni, and Fabrizio Silvestri. What are you sinking? a geometric approach on attention sink. *arXiv preprint arXiv:2508.02546*, 2025.

[509] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[510] Yalin E. Sagduyu, Sennur Ulukus, and Aylin Yener. Task-oriented communications for nextg: End-to-end deep learning and ai security aspects. *IEEE Wireless Communications*, 30(3):52–60, 2023.

[511] Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

[512] M. Sala. The EB II 'metallic ware' from Tell el-Far 'ah North (West Bank): typology, technology and petrography of a ceramic industry of the central hill country. *Levant*, 55(2):146–171, 2023.

[513] Mohammad Samragh, Mehrdad Farajtabar, Sachin Mehta, Raviteja Vemulapalli, Fartash Faghri, Devang Naik, Oncel Tuzel, and Mohammad Rastegari. Weight subcloning: direct initialization of transformers using larger pretrained ones. 2023.

[514] Mohammad Samragh, Mehrdad Farajtabar, Sachin Mehta, Raviteja Vemulapalli, Fartash Faghri, Devang Naik, Oncel Tuzel, and Mohammad Rastegari. Weight subcloning: direct initialization of transformers using larger pretrained ones. 2023.

[515] Sreetama Sarkar, Souvik Kundu, Kai Zheng, and Peter A. Beerel. Block selective reprogramming for on-device training of vision transformers. 2024.

[516] Simone Sarti, Eugenio Lomurno, and Matteo Matteucci. Anticipate, ensemble and prune: Improving convolutional neural networks via aggregated early exits. *Procedia Computer Science*, 222:519–528, 2023.

[517] S Scardapane, A Baiocchi, A Devoto, V Marsocci, P Minervini, and J Pomponi. Conditional computation in neural networks: Principles and research trends. *Intelligenza Artificiale*, 18:175–190, 2024.

[518] Simone Scardapane, Alessandro Baiocchi, Alessio Devoto, Valerio Marsocci, Pasquale Minervini, and Jary Pomponi. Conditional computation in neural networks: Principles and research trends. *Intelligenza Artificiale*, 18(1).

[519] Simone Scardapane, Alessandro Baiocchi, Alessio Devoto, Valerio Marsocci, Pasquale Minervini, and Jary Pomponi. Conditional computation in neural networks: Principles and research trends. *Intelligenza Artificiale*, 18(1):175–190, July 2024.

[520] Simone Scardapane, Alessandro Baiocchi, Alessio Devoto, Valerio Marsocci, Pasquale Minervini, and Jary Pomponi. Conditional computation in neural networks: Principles and research trends. *Intelligenza Artificiale*, 18(1):175–190, July 2024.

[521] Simone Scardapane, Danilo Comminiello, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini. Differentiable branching in deep networks for fast inference. In *Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4167–4171. IEEE, 2020.

[522] Simone Scardapane, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini. Why should we add early exits to neural networks? *Cognitive Computation*, 12(5):954–966, 2020.

[523] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

[524] Patrick Schramowski, Cigdem Turan, Sophie F. Jentzsch, Constantin A. Rothkopf, and Kristian Kersting. BERT has a moral compass: Improvements of ethical and moral values of machines. *CoRR*, abs/1912.05238, 2019.

[525] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

[526] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Commun. ACM*, 63(12):54–63, 2020.

[527] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, October 2019.

[528] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proceedings of the 5th International Conference on Learning Representations*, 2017. https://doi.org/10.48550/arXiv.1701.06538.

[529] Noam Shazeer. Fast transformer decoding: One write-head is all you need. 2019.

[530] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv*, 2019.

[531] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proceedings of the 2017 International Conference on Learning Representations (ICLR)*, 2017.

[532] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

[533] Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. Mixture models for diverse machine translation: Tricks of the trade. In *International Conference on Machine Learning*, pages 5719–5728. PMLR, 2019.

[534] Luohe Shi, Hongyi Zhang, Yao Yao, Zuchao Li, and Hai Zhao. Keep the cost down: A review on methods to optimize llm' s kv-cache consumption. *First Conference on Language Modeling (COLM)*, 2024.

[535] Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. Trusting your evidence: Hallucinate less with context-aware decoding. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors,

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Short Papers, NAACL 2024, Mexico City, Mexico, June 16-21, 2024, pages 783–791. Association for Computational Linguistics, 2024.

[536] J. Shlomi, P. Battaglia, and J.-R. Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2, 2020. https://doi.org/10.1088/2632-2153/abbf9a.

[537] Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. Lora vs full fine-tuning: An illusion of equivalence. 2024.

[538] Shashwat Singh, Shauli Ravfogel, Jonathan Herzig, Roee Aharoni, Ryan Cotterell, and Ponnurangam Kumaraguru. Mimic: Minimally modified counterfactuals in the representation space. *CoRR*, abs/2402.09631, 2024.

[539] P. Siozos, N. Hausmann, M. Holst, and D. Anglos. Application of laser-induced breakdown spectroscopy and neural networks on archaeological human bones for the discrimination of distinct individuals. *Journal of Archaeological Science: Reports*, 35:102769, 2021.

[540] Nicolas Skatchkovsky and Osvaldo Simeone. Optimizing pipelined computation and communication for latency-constrained edge learning. *IEEE Communications Letters*, 23(9):1542–1546, 2019.

[541] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

[542] Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

[543] Zhuoran Song, Yihong Xu, Zhezhi He, Li Jiang, Naifeng Jing, and Xiaoyao Liang. Cp-vit: Cascade vision transformer pruning via progressive sparsity prediction. *arXiv preprint arXiv:2203.04570*, 2022.

[544] Balasubramaniam Srinivasan and Bruno Ribeiro. On the equivalence between positional node embeddings and structural graph representations. In *8th International Conference on Learning Representations*, 2019. https://doi.org/10.48550/arXiv.1910.00452.

[545] Konrad Staniszewski, Szymon Tworkowski, Yu Zhao, Sebastian Jaszczur, Henryk Michalewski, Lukasz Kuci'nski, and Piotr Milo's. Structured packing in llm training improves long context utilization. *ArXiv*, abs/2312.17296, 2023.

[546] R. Staszewski and J. Chwastowski. Transport simulation and diffractive event reconstruction at the lhc. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 609(2):136–141, 2009. https://doi.org/10.1016/j.nima.2009.08.023.

[547] M. Steiner and A. E. Killebrew. *The Oxford Handbook of the Archaeology of the Levant: c. 8000-332 BCE*. Oxford Handbooks, 2014.

[548] StepFun, :, Bin Wang, Bojun Wang, Changyi Wan, Guanzhe Huang, Hanpeng Hu, Haonan Jia, Hao Nie, Mingliang Li, Nuo Chen, Siyu Chen, Song Yuan, Wuxun Xie, Xiaoniu Song, Xing Chen, Xingping Yang, Xuelin Zhang, Yanbo Yu, Yaoyu Wang, Yibo Zhu, Yimin Jiang, Yu Zhou, Yuanwei Lu, Houyi Li, Jingcheng Hu, Ka Man Lo, Ailin Huang, Binxing Jiao, Bo Li, Boyu Chen, Changxin Miao, Chang Lou, Chen Hu, Chen Xu, Chenfeng Yu, Chengyuan Yao, Daokuan Lv, Dapeng Shi, Deshan Sun, Ding Huang, Dingyuan Hu, Dongqing Pang, Enle Liu, Fajie Zhang, Fanqi Wan, Gulin Yan, Han Zhang, Han Zhou, Hanghao Wu, Hangyu Guo, Hanqi Chen, Hanshan Zhang, Hao Wu, Haocheng Zhang, Haolong Yan, Haoran Lv, Haoran Wei, Hebin Zhou, Heng Wang, Heng Wang, Hongxin Li, Hongyu Zhou, Hongyuan Wang, Huiyong Guo, Jia Wang, Jiahao Gong, Jialing Xie, Jian Zhou, Jianjian Sun, Jiaoren Wu, Jiaran Zhang, Jiayu Liu, Jie Cheng, Jie Luo, Jie Yan, Jie Yang, Jieyi Hou, Jinguang Zhang, Jinlan Cao, Jisheng Yin, Junfeng Liu, Junhao Huang, Junzhe Lin, Kaijun Tan, Kaixiang Li, Kang An, Kangheng Lin, Kenkun Liu, Lei Yang, Liang Zhao, Liangyu Chen, Lieyu Shi, Liguo Tan, Lin Lin, Lin Zhang, Lina Chen, Liwen Huang, Liying Shi, Longlong Gu, Mei Chen, Mengqiang Ren, Ming Li, Mingzhe Chen, Na Wang, Nan Wu, Qi Han, Qian Zhao, Qiang Zhang, Qianni Liu, Qiaohui Chen, Qiling Wu, Qinglin He, Qinyuan Tan, Qiufeng Wang, Qiuping Wu, Qiuyan Liang, Quan Sun, Rui Li, Ruihang Miao, Ruosi Wan, Ruyan Guo, Shangwu Zhong, Shaoliang Pang, Shengjie Fan, Shijie Shang, Shilei Jiang, Shiliang Yang, Shiming Hao, Shuli Gao, Siming Huang, Siqi Liu, Tiancheng Cao, Tianhao Cheng, Tianhao Peng, Wang You, Wei Ji, Wen Sun, Wenjin Deng, Wenqing He, Wenzhen Zheng, Xi Chen, Xiangwen Kong, Xianzhen Luo, Xiaobo Yang, Xiaojia Liu, Xiaoxiao Ren, Xin Han, Xin Li, Xin Wu, Xu Zhao, Yanan Wei, Yang Li, Yangguang Li, Yangshijie Xu, Yanming Xu, Yaqiang Shi, Yeqing Shen, Yi Yang, Yifei Yang, Yifeng Gong, Yihan Chen, Yijing Yang, Yinmin Zhang, Yizhuang Zhou, Yuanhao Ding, Yuantao Fan, Yuanzhen Yang, Yuchu Luo, Yue Peng, Yufan Lu, Yuhang Deng, Yuhe Yin, Yujie Liu, Yukun Chen, Yuling Zhao, Yun Mou, Yunlong Li, Yunzhou Ju, Yusheng Li, Yuxiang Yang, Yuxiang Zhang, Yuyang Chen, Zejia Weng, Zhe Xie, Zheng Ge, Zheng Gong, Zhenyi Lu, Zhewei Huang, Zhichao Chang, Zhiguo Huang, Zhirui Wang, Zidong Yang, Zili Wang, Ziqi Wang, Zixin Zhang, Binxing Jiao, Daxin Jiang, Heung-Yeung Shum, and Xiangyu Zhang. Step-3 is large yet affordable: Model-system co-design for cost-effective decoding, 2025.

[549] Emilio Calvanese Strinati and Sergio Barbarossa. 6G networks: Beyond Shannon towards semantic and goal-oriented communications. *Computer Networks*, 190:107930, 2021.

[550] Emilio Calvanese Strinati, Paolo Di Lorenzo, Vincenzo Sciancalepore, Adnan Aijaz, Marios Kountouris, Deniz Gündüz, Petar Popovski, Mohamed Sana, Photios A Stavrou, Beatriz Soret, et al. Goal-oriented and semantic communication in 6G AI-native networks: The 6G-GOALS approach. 2024.

[551] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[552] Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, et al. Self-conditioned embedding diffusion for text generation. *arXiv preprint arXiv:2211.04236*, 2022.

[553] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023.

[554] Zhaochen Su, Jun Zhang, Xiaoye Qu, Tong Zhu, Yanshu Li, Jiashuo Sun, Juntao Li, Min Zhang, and Yu Cheng. Conflictbank: A benchmark for evaluating the influence of knowledge conflicts in llm. *arXiv preprint arXiv:2408.12076*, 2024.

[555] Nishant Subramani, Nivedita Suresh, and Matthew E. Peters. Extracting latent steering vectors from pretrained language models. In *ACL (Findings)*, pages 566–581. Association for Computational Linguistics, 2022.

[556] Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive activations in large language models, 2024.

[557] Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

[558] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. 2022.

[559] F Szatkowski, P Będkowski, A Devoto, J Dubiński, P Minervini, et al. Universal properties of activation sparsity in modern large language models. arXiv preprint arXiv:2509.00454, 2025.

[560] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[561] Shawn Tan, Yikang Shen, Zhenfang Chen, Aaron Courville, and Chuang Gan. Sparse universal transformer. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 169–179. Association for Computational Linguistics, December 2023.

[562] Shengkun Tang, Yaqing Wang, Zhenglun Kong, Tianchi Zhang, Yao Li, Caiwen Ding, Yanzhi Wang, Yi Liang, and Dongkuan Xu. You need multiple exiting: Dynamic early exiting for accelerating unified vision language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781–10791, 2023.

[563] Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35:21831–21843, 2022.

[564] Gemini Team. Gemini diffusion. https://deepmind.google/models/gemini-diffusion/, 2025.

[565] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *Proceedings of the 2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

[566] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024.

[567] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024.

[568] S. Thais, P. Calafiura, G. Chachamis, G. DeZoort, J. Duarte, S. Ganguly, M. Kagan, D. Murnane, M. Neubauer, and K. Terao. Graph neural networks in particle physics: Implementations, innovations, and challenges, 2022. https://doi.org/10.48550/arXiv.2203.12852.

[569] Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19, 2019.

[570] Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[571] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers with distillation through attention. 139:10347–10357, July 2021.

[572] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *European Conf. on Computer Vision*, pages 516–533, 2022.

[573] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *European Conference on Computer Vision*, pages 516–533. Springer, 2022.

[574] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[575] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[576] Marcos Treviso, António Góis, Patrick Fernandes, Erick Fonseca, and Andre Martins. Predicting attention sparsity in transformers. In Andreas Vlachos, Priyanka Agrawal, André Martins, Gerasimos Lampouras, and Chunchuan Lyu, editors, *Proceedings of the Sixth Workshop on Structured Prediction for NLP*, pages 67–81. Association for Computational Linguistics, May 2022.

[577] Ø. D. Trier, D. C. Cowley, and A. U. Waldeland. Using deep neural networks on airborne laser scanning data: Results from a case study of semi-automatic mapping of archaeological topography on Arran, Scotland. *Archaeological Prospection*, 26(2):165–175, 2019.

[578] M. Troiano, E. Nobile, F. Mangini, M. Mastrogiuseppe, C. Conati Barbaro, and F. Frezza. A comparative analysis of the bayesian regularization and levenberg–marquardt training algorithms in neural networks for small datasets: A metrics prediction of neolithic laminar artefacts. *Information*, 15(5), 2024.

[579] V. Tumolo and K. Badreshany. The 'Combed Ware' storage and transport vessels from Khirbet ez-Zeraqon: a reappraisal of the EB II-III evidence in light of recent studies. *Journal of Ancient Egyptian Interconnections*, 37:301–324, 2023.

[580] Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *CoRR*, abs/2308.10248, 2023.

[581] Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *CoRR*, abs/2308.10248, 2023.

[582] Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems*, 36, 2024.

[583] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5998–6008, 2017. https://doi.org/10.48550/arXiv.1706.03762.

[584] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, L ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett,

editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[585] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[586] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[587] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[588] A. Verdone, A. Devoto, C. Sebastiani, J. Carmignani, M. D'Onofrio, S. Giagu, S. Scardapane, and M. Panella. Enhancing High-Energy Particle Physics Collision Analysis through Graph Data Attribution Techniques, 2024. https://doi.org/10.48550/arXiv.2407.14859.

[589] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020.

[590] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 2320–2329, 2020.

[591] Thomas Verelst and Tinne Tuytelaars. Segblocks: Block-based dynamic resolution networks for real-time segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2400–2411, 2022.

[592] Tu Vu, Aditya Barua, Brian Lester, Daniel Cer, Mohit Iyyer, and Noah Constant. Overcoming catastrophic forgetting in zero-shot cross-lingual generation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9279–9300, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[593] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.

[594] Lingchen Wang, Xuanyi Huang, Jie Huang, Hongsheng Li, and Lei Zhang. Adavit: Adaptive vision transformers for efficient image recognition. pages 12309–12318, 2021.

[595] Lingyi Wang, Wei Wu, Fuhui Zhou, Zhaohui Yang, and Zhijin Qin. Adaptive resource allocation for semantic communication networks. 2023.

[596] Meiqi Wang, Jianqiao Mo, Jun Lin, Zhongfeng Wang, and Li Du. Dynexit: A dynamic early-exit strategy for deep residual networks. In *Proceedings of the 2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 178–183. IEEE, 2019.

[597] X. Wang and Y. Li. Harmonized dense knowledge distillation training for multi-exit architectures. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:10218–10226, 2021.

[598] Xin Wang, Yujia Luo, Daniel Crankshaw, Alexey Tumanov, Fisher Yu, and Joseph E Gonzalez. Idk cascades: Fast deep learning by learning not to overthink.

[599] Yike Wang, Shangbin Feng, Heng Wang, Weijia Shi, Vidhisha Balachandran, Tianxing He, and Yulia Tsvetkov. Resolving knowledge conflicts in large language models. *CoRR*, abs/2310.00935, 2023.

[600] Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. *arXiv preprint arXiv:2509.06949*, 2025.

[601] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. *Advances in Neural Information Processing Systems*, 34:11960–11973, 2021.

[602] Zhiyuan Wang and Haisheng Tan. Towards efficient inference on mobile device via pruning. In *2024 10th International Conference on Big Data Computing and Communications (BigCom)*, pages 26–33, 2024.

[603] H. Weiss. The northern levant during the intermediate bronze age. In Margreet L. Steiner and Ann E. Killebrew, editors, *The Oxford Handbook of the Archaeology of the Levant: c. 8000–332 BCE*, pages 401–412. Oxford University Press, Oxford, 2013.

[604] Yuetian Weng, Mingfei Han, Haoyu He, Xiaojun Chang, and Bohan Zhuang. Longvlm: Efficient long video understanding via large language models. In *European Conference on Computer Vision*, pages 453–470. Springer, 2024.

[605] I. K. Whitbrad. The characterisation of argillaceous inclusions in ceramic thin sections. *Archaeometry*, 28(1):79–88, 1986.

[606] I. K. Whitbread and A. Hunt. *Fabric description of archaeological ceramics. The Oxford handbook of archaeological ceramic analysis.* Oxford University Press, 2017.

[607] I. K. Whitebread. *Greek transport amphorae. A petrological and archaeological study.* British School of Athens., 1995.

[608] Ross Wightman. Pytorch image models. *GitHub repository*, 2019.

[609] B Wójcik, A Devoto, K Pustelnik, P Minervini, and S Scardapane. Adaptive modular computation: Granular conditional computation for efficient inference. In *AAAI 2025 - Proceedings of the Thirty-ninth AAAI Conference on Artificial ...*, 2025.

[610] Bartosz Wójcik, Alessio Devoto, Karol Pustelnik, Pasquale Minervini, and Simone Scardapane. Adaptive computation modules: Granular conditional computation for efficient inference. *arXiv preprint arXiv:2312.10193*, 2023.

[611] Maciej Wołczyk, Bartosz Wójcik, Klaudia Bałazy, Igor T Podolak, Jacek Tabor, Marek Śmieja, and Tomasz Trzcinski. Zero time waste: Recycling predictions in early exit neural networks. *Advances in Neural Information Processing Systems*, 34:2516–2528, 2021.

[612] Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion llm. *arXiv preprint arXiv:2509.26328*, 2025.

[613] Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. *arXiv preprint arXiv:2505.22618*, 2025.

[614] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020.

[615] Haotian Wu, Yulin Shao, Chenghong Bian, Krystian Mikolajczyk, and Deniz Gündüz. Deep joint source-channel coding for adaptive image transmission over mimo channels. 2023.

[616] Yuxiang Wu, Yu Zhao, Baotian Hu, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. An efficient memory-augmented transformer for knowledge-intensive NLP tasks. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5184–5196. Association for Computational Linguistics, 2022.

[617] Wu, Haotian and Shao, Yulin and Bian, Chenghong and Mikolajczyk, Krystian and Gündüz, Deniz. Deep joint source-channel coding for adaptive image transmission over mimo channels. *IEEE Transactions on Wireless Communications*, 2024.

[618] Bartosz Wójcik, Alessio Devoto, Karol Pustelnik, Pasquale Minervini, and Simone Scardapane. Adaptive computation modules: Granular conditional computation for efficient inference. 2023.

[619] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2(2):109–127, 2021. https://doi.org/10.1109/TAI.2021.3076021.

[620] Wenhan Xia, Hongxu Yin, Xiaoliang Dai, and Niraj K Jha. Fully dynamic inference with deep neural networks. *IEEE Transactions on Emerging Topics in Computing*, 10(2):962–972, 2021.

[621] Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *Internation Conference on Learning Represenation*, 2024.

[622] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *Internation Conference on Learning Represenations*, 2023.

[623] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024.

[624] Huiqiang Xie, Zhijin Qin, Geoffrey Ye Li, and Biing-Hwang Juang. Deep learning enabled semantic communication systems. *IEEE Transactions on Signal Processing*, 69:2663–2675, 2021.

[625] Huiqiang Xie, Zhijin Qin, Geoffrey Ye Li, and Biing-Hwang Juang. Deep learning enabled semantic communication systems. *IEEE Transactions on Signal Processing*, 69:2663–2675, 2021.

[626] Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[627] Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *ICLR*. OpenReview.net, 2024.

[628] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251. Association for Computational Linguistics, July 2020.

[629] Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. BERxiT: Early exiting for BERT with better fine-tuning and extension to regression. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 91–104, Online, April 2021. Association for Computational Linguistics.

[630] Yi Xin, Siqi Luo, Haodi Zhou, Junlong Du, Xiaohong Liu, Yue Fan, Qing Li, and Yuntao Du. Parameter-efficient fine-tuning for pre-trained vision models: A survey. 2024.

[631] Qunliang Xing, Mai Xu, Tianyi Li, and Zhenyu Guan. Early exit or not: Resource-efficient blind quality enhancement for compressed images. In *European Conference on Computer Vision*, pages 275–292. Springer, 2020.

[632] Jialong Xu, Tze-Yang Tung, Bo Ai, Wei Chen, Yuxuan Sun, and Deniz Deniz Gündüz. Deep joint source-channel coding for semantic communications. *IEEE Communications Magazine*, 61(11):42–48, 2023.

[633] Rongwu Xu, Zehan Qi, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. Knowledge conflicts for llms: A survey. *arXiv preprint arXiv:2403.08319*, 2024.

[634] Wei Xu, Zhaohui Yang, Derrick Wing Kwan Ng, Marco Levorato, Yonina C Eldar, and Mérouane Debbah. Edge learning for b5g networks with distributed signal processing: Semantic communication, edge computing, and wireless sensing. *IEEE journal of selected topics in signal processing*, 17(1):9–39, 2023.

[635] Xuwei Xu, Sen Wang, Yudong Chen, Yanping Zheng, Zhewei Wei, and Jiajun Liu. Gtp-vit: Efficient vision transformers via graph-based token propagation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 86–95, January 2024.

[636] Zichuan Xu, Liqian Zhao, Weifa Liang, Omer F. Rana, Pan Zhou, Qiufen Xia, Wenzheng Xu, and Guowei Wu. Energy-aware inference offloading for dnn-driven applications in mobile edge clouds. *IEEE Transactions on Parallel and Distributed Systems*, 32(4):799–814, 2021.

[637] Fuzhao Xue, Valerii Likhosherstov, Anurag Arnab, Neil Houlsby, Mostafa Dehghani, and Yang You. Adaptive computation with elastic input sequence. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[638] Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. Openmoe: An early effort on open mixture-of-experts language models. *arXiv preprint arXiv:2402.01739*, 2024.

[639] Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search, 2025.

[640] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv*, 2025.

[641] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3318–3327, 2019.

[642] Ke Yang, Sixian Wang, Jincheng Dai, Xiaoqi Qin, Kai Niu, and Ping Zhang. Swinjscc: Taming swin transformer for deep joint source-channel coding. *IEEE Transactions on Cognitive Communications and Networking*, 2024.

[643] Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025.

[644] Mingyu Yang and Hun-Seok Kim. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5193–5197, 2022.

[645] S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, and F. Shen. Image Data Augmentation for Deep Learning: A Survey., 2023.

[646] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics, 2018.

[647] Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *International Conference on Learning Representations*, 2025.

[648] Jiacheng Ye, Shansan Gong, Liheng Chen, Lin Zheng, Jiahui Gao, Han Shi, Chuan Wu, Xin Jiang, Zhenguo Li, Wei Bi, et al. Diffusion of thought: Chain-of-thought reasoning in diffusion language models. *Advances in Neural Information Processing Systems*, 37:105345–105374, 2024.

[649] Jiacheng Ye, Zhenyu Wu, Jiahui Gao, Zhiyong Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Implicit search via discrete diffusion: A study on chess. *International Conference on Learning Representations*, 2025.

[650] Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv*, 2025.

[651] Hongxu Yin, Arash Vahdat, Jose Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[652] Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10809–10818, 2022.

[653] Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. pages 10809–10818, 2022.

[654] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 9240–9251, 2019. https://doi.org/10.48550/arXiv.1903.03894.

[655] G. Yu, Z. Jin, L. Chen, F. Wang, X. Wang, X Wu, A. Fan, and Q. Xia. Analyzing the earliest Chinese proto-porcelain: Study on the materials from Liaotianjianshan kiln sites, Dehua County, Fujian Province (China). *Ceramics International*, 44(17):21648–21655, 2018.

[656] Xiaoyu Yue, Shuyang Sun, Zhanghui Kuang, Meng Wei, Philip HS Torr, Wayne Zhang, and Dahua Lin. Vision transformer with progressive sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 387–396, 2021.

[657] Seniha Esen Yuksel, Joseph N Wilson, and Paul D Gader. Twenty years of mixture of experts. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1177–1193, 2012.

[658] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[659] Zeyu Yun, Yubei Chen, Bruno Olshausen, and Yann LeCun. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. In Eneko Agirre, Marianna Apidianaki, and Ivan Vulić, editors, *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 1–10, Online, June 2021. Association for Computational Linguistics.

[660] Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*, 2023.

[661] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

[662] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. pages 12104–12113, 2022.

[663] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? 2022.

[664] Guangyi Zhang, Qiyu Hu, Zhijin Qin, Yunlong Cai, and Guanding Yu. A unified multi-task semantic communication system with domain adaptation. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 3971–3976. IEEE, 2022.

[665] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[666] J. Zhang, I. Fang, H. Wu, A. Kaushik, A. Rodriguez, H. Zhao, J. Zhang, Z. Zheng, R. Iovita, and C. Feng. Luwa dataset: Learning lithic use-wear analysis on microscopic images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22573, 2024.

[667] Michael J. Q. Zhang and Eunsol Choi. Situatedqa: Incorporating extra-linguistic contexts into QA. In *EMNLP (1)*, pages 7371–7387. Association for Computational Linguistics, 2021.

[668] Milin Zhang, Mohammad Abdi, Venkat R. Dasari, and Francesco Restuccia. Semantic edge computing and semantic communications in 6g networks: A unifying survey and research challenges. 2024.

[669] Renrui Zhang, Rongyao Fang, Wei Zhang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free clip-adapter for better vision-language modeling. 2021.

[670] Wenyu Zhang, Haijun Zhang, Hui Ma, Hua Shao, Ning Wang, and Victor C. M. Leung. Predictive and adaptive deep coding for wireless image transmission in semantic communication. *IEEE Transactions on Wireless Communications*, 22(8):5486–5501, 2023.

[671] Xiaofeng Zhang, Yikang Shen, Zeyu Huang, Jie Zhou, Wenge Rong, and Zhang Xiong. Mixture of attention heads: Selecting attention heads per token, 2022.

[672] Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞Bench: Extending long context evaluation beyond 100K tokens. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[673] Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. MoEfication: Transformer feed-forward layers are mixtures of experts. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 877–890, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[674] Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts, 2022.

[675] Zhengyan Zhang, Zhiyuan Zeng, Yankai Lin, Chaojun Xiao, Xiaozhi Wang, Xu Han, Zhiyuan Liu, Ruobing Xie, Maosong Sun, and Jie Zhou. Emergent modularity in pre-trained transformers. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4066–4083. Association for Computational Linguistics, July 2023.

[676] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[677] Wanru Zhao, Vidit Khazanchi, Haodi Xing, Xuanli He, Qiongkai Xu, and Nicholas Donald Lane. Attacks on third-party apis of large language models. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*, 2024.

[678] Y Zhao, A Devoto, G Hong, X Du, AP Gema, H Wang, X He, KF Wong, et al. Steering knowledge selection behaviours in llms via sae-based representation engineering. In *NAACL 2025 - Nations of the Americas Chapter of the Association for...*, 2025.

[679] Yu Zhao, Xiaotang Du, Giwon Hong, Aryo Pradipta Gema, Alessio Devoto, Hongru Wang, Xuanli He, Kam-Fai Wong, and Pasquale Minervini. Analysing the residual stream of language models under knowledge conflicts. *CoRR*, abs/2410.16090, 2024.

[680] Yu Zhao, Yuanbin Qu, Konrad Staniszewski, Szymon Tworkowski, Wei Liu, Piotr Miłoś, Yuxiang Wu, and Pasquale Minervini. Analysing the impact of sequence composition on language model pre-training. *arXiv preprint arXiv:2402.13991*, 2024.

[681] Yu Zhao, Yuanbin Qu, Konrad Staniszewski, Szymon Tworkowski, Wei Liu, Piotr Miłoś, Yuxiang Wu, and Pasquale Minervini. Analysing the impact of sequence composition on language model pre-training. *arXiv preprint arXiv:2402.13991*, 2024.

[682] Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir R. Radev. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5905–5921. Association for Computational Linguistics, 2021.

[683] X. Zhong, B. Gallagher, S. Liu, B. Kailkhura, A. Hiszpanski, and T. Y. J. Han. Explainable machine learning in materials science. *Computational Materials*, 8, 2022.

[684] Zexuan Zhong, Ziqing Huang, Alexander Wettig, and Danqi Chen. Poisoning retrieval corpora by injecting adversarial passages. *arXiv preprint arXiv:2310.19156*, 2023.

[685] Qingyang Zhou, Rongpeng Li, Zhifeng Zhao, Chenghui Peng, and Honggang Zhang. Semantic communication with adaptive universal transformer. *IEEE Wireless Communications Letters*, 11(3):453–457, 2021.

[686] Qingyang Zhou, Rongpeng Li, Zhifeng Zhao, Yong Xiao, and Honggang Zhang. Adaptive bit rate control in semantic communication with incremental knowledge-based harq. *IEEE Open Journal of the Communications Society*, 3:1076–1089, 2022.

[687] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341, 2020.

[688] Y. Zhou, S. Booth, M. T. Ribeiro, and J. Shah. Do feature attribution methods correctly attribute features? In *Proceedings in the 36th AAAI Conference on Artificial Intelligence*, pages 9623–9633. AAAI Press, 2022. https://doi.org/10.48550/arXiv.2104.14403.

[689] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

[690] Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, Hongrui Guo, Jiaqi Hu, Wentao Ye, Tieyuan Chen, Chenchen Li, Chengfu Tang, Haibo Feng, Jun Hu, Jun Zhou, Xiaolu Zhang, Zhenzhong Lan, Junbo Zhao, Da Zheng, Chongxuan Li, Jianguo Li, and Ji-Rong Wen. Llada-moe: A sparse moe diffusion language model, 2025.

[691] Jinguo Zhu, Xizhou Zhu, Wenhai Wang, Xiaohua Wang, Hongsheng Li, Xiaogang Wang, and Jifeng Dai. Uni-perceiver-moe: Learning sparse generalist models with conditional moes. *Advances in Neural Information Processing Systems*, 35:2664–2678, 2022.

[692] Mingjian Zhu, Kai Han, Enhua Wu, Qiulin Zhang, Ying Nie, Zhenzhong Lan, and Yunhe Wang. Dynamic resolution network. *Advances in Neural Information Processing Systems*, 34:27319–27330, 2021.

[693] Wei Zhu. LeeBERT: Learned early exit for BERT with cross-level optimization. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2968–2980, Online, August 2021. Association for Computational Linguistics.

[694] Yitao Zhu, Zhenrong Shen, Zihao Zhao, Sheng Wang, Xin Wang, Xiangyu Zhao, Dinggang Shen, and Qian Wang. Melo: Low-rank adaptation is better than fine-tuning for medical image diagnosis. 2023.

[695] Zeyuan Allen Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023.

[696] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi : Plug-and-play 2bit kv cache quantization with streaming asymmetric quantization. *ICML*, 2023.

[697] Chen Ziwen, Kaushik Patnaik, Shuangfei Zhai, Alvin Wan, Zhile Ren, Alexander G Schwing, Alex Colburn, and Li Fuxin. Autofocusformer: Image segmentation off the grid. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18227–18236, 2023.

[698] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

[699] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to AI transparency. *CoRR*, abs/2310.01405, 2023.

[700] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to AI transparency. *CoRR*, abs/2310.01405, 2023.

[701] Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. Poisonedrag: Knowledge poisoning attacks to retrieval-augmented generation of large language models. *arXiv preprint arXiv:2402.07867*, 2024.

[702] Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Tuo Zhao, and Jianfeng Gao. Taming sparsely activated transformer with stochastic experts. In *Proceedings of the 2022 International Conference on Learning Representations (ICLR)*, 2022.

[703] Adrian Lańcucki, Konrad Staniszewski, Piotr Nawrot, and Edoardo M. Ponti. Inference-time hyper-scaling with kv cache compression. *arXiv*, 2025.