A survey on efficient vision transformers: algorithms, techniques, and performance benchmarking

Lorenzo Papa, *Student Member, IEEE,* Paolo Russo, Irene Amerini, *Member, IEEE,* and Luping Zhou, *Senior Member, IEEE*

Abstract—Vision Transformer (ViT) architectures are becoming increasingly popular and widely employed to tackle computer vision applications. Their main feature is the capacity to extract global information through the self-attention mechanism, outperforming earlier convolutional neural networks. However, ViT deployment and performance have grown steadily with their size, number of trainable parameters, and operations. Furthermore, self-attention's computational and memory cost quadratically increases with the image resolution. Generally speaking, it is challenging to employ these architectures in real-world applications due to many hardware and environmental restrictions, such as processing and computational capabilities. Therefore, this survey investigates the most efficient methodologies to ensure sub-optimal estimation performances. More in detail, four efficient categories will be analyzed: compact architecture, pruning, knowledge distillation, and quantization strategies. Moreover, a new metric called *Efficient Error Rate* has been introduced in order to normalize and compare models' features that affect hardware devices at inference time, such as the number of parameters, bits, FLOPs, and model size. Summarizing, this paper firstly mathematically defines the strategies used to make Vision Transformer efficient, describes and discusses state-of-the-art methodologies, and analyzes their performances over different application scenarios. Toward the end of this paper, we also discuss open challenges and promising research directions.

Index Terms—Computer vision, computational efficiency, vision transformer

1 Introduction

Artificial intelligence (AI) solutions based on deep learning (DL) infrastructures are becoming increasingly popular in a variety of everyday life and industrial application scenarios, such as chat-bots and perception systems [13, 44, 14, 45, 80]. Those tasks are usually based on neural language processing (NLP) and computer vision (CV) solutions, specifically focusing on text and image analysis. Although such algorithms have usually been developed through convolutional neural networks (CNN) models, i.e., architectures consisting of convolutional operations used to extract information at different scales, recently, new families of neural networks such as [62, 18, 59] have been proposed. These methodologies exhibit exceptional performance in AI applications and consistently push their limits. They have in common the self-attention mechanism, which simultaneously extracts specific information from each input data, including text prompts and pixels, while also considering their inter-relationships for a sensible improvement of the simple translation-invariant property of the convolution operator. Generally speaking, the attention mechanism has been introduced by Bahdanau et al. [1] (2014) in order to address the bottleneck problem that arises in encoder-decoder architectures to flexibly translate the most relevant information of the encoded input sequences into the decoder part. In NLP tasks, this issue is especially true for lengthy and/or sophisticated sequences, while in CV-dense prediction applications such as semantic segmentation and depth estimation, this technique will lead to superior reconstruction capabilities of the model's outputs, as shown in [9, 78, 51]. Furthermore, the attention mechanism is then reformulated by Vaswani et al. [62] (2017) in order to extract intrinsic features while showing the self-attention potential in order to capture long-range dependencies.

Since their first developments, transformer models have usually been applied to NLP scenarios such as language-to-language translation tasks [50, 2]. Subsequently, inspired by the remarkable performances achieved by the global receptive field of transformer architectures in CV tasks, Dai et al. [12] (2017) introduce the deformable convolutions in order to overcome the fixed geometric structures of CNN modules. Similarly, Wang et al. [67] (2018) present non-local operation techniques for capturing long-range dependencies, showing that non-local neural networks outperform well-known 2D and 3D convolutional architectures for video classification tasks. Moreover, Zhang et al. [82] (2020) propose to model long-rate dependencies via graph

- *L. Papa is the corresponding author. The work has been developed during the visiting research period at The University of Sydney, Australia, NSW, 2006.*
- *L. Papa, P. Russo and I. Amerini are with the Department of Computer, Control and Management Engineering, Sapienza University of Rome, Italy, IT, 00185. E-mail: [papa, paolo.russo, amerini]@diag.uniroma1.it*
- *L. Papa and L. Zhou are with the School of Electrical and Information Engineering, Faculty of Engineering, The University of Sydney, Australia, NSW, 2006. E-mail: [lorenzo.papa, luping.zhou]@sydney.edu.au*

Manuscript received April 19, 2005; revised August 26, 2015.
L. Zhou would like to acknowledge the The USyd-Fudan BISA Flagship Research Program (2023), I. Amerini would like to acknowledge SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU, Sapienza University of Rome project 2022–2024 "EV2" (003_009_22), and L. Papa would like to acknowledge project 2022–2023 "RobFastMDE".

convolutional neural network. The authors introduce a dynamic graph message passing network (DGMN) which is able to achieve substantial improvements with respect to dynamic convolution operations while using fewer MAC operations. These results are achieved thanks to the larger receptive field and less-redundant information captured by graph nodes. Despite these solutions, which demonstrate the ability to improve CNN models, Dosovitskiy et al. [16] (2021) propose the first solution in order to apply the self-attention to bi-dimensional signals, namely vision transformers (ViT). The authors design a general backbone where the input image is divided into fixed-size patches in order to apply the standard self-attention. This preliminary work exhibits the capabilities and performances of transformers also in the CV research field. However, one of the major challenges for ViT is the computational cost of their key element, i.e., the self-attention itself; this behavior is particularly true for high-resolution and dense prediction tasks [66, 10, 78, 51]. In particular, self-attention's computational and memory cost increases quadratically with the image resolution. Moreover, the Softmax operation computed in the attention block makes these structures computationally demanding for edge and low-resource devices. These demanding hardware requirements provide significant hurdles for ViT models to infer on resource-constrained devices such as embedded devices and autonomous systems. Furthermore, to provide a high-quality user experience, real-time computer vision systems incorporating transformer-based models must fulfill low latency requirements. Consequently, due to the growing development of novel ViT architectures with improved estimation performances (which comes at the expense of elevated computational costs) and the need for resource-constrained devices in real-world AI tasks, this survey aims to give an in-depth analysis of the most recent solutions in order to design **efficient ViT models**.

Generally speaking, previously proposed surveys by Han et al. [19] (2022), Tay et al. [58] (2022), and Khan et al. [31] (2022) mainly focus on a general overview of transformers models from both NLP and CV tasks and their various application scenarios. Specifically, those earlier studies exclusively rely on small subsections, future investigations, or a limited amount of analyzed methodologies related to the efficient ViT strategies. Furthermore, several works have been recently conducted (2023) in response to the rising need for more accurate models capable of performing tasks in real-world settings, resulting in significant progress in the research field. In contrast to previous surveys, Zhuang et al. [85] (2023) recently presented a systematic overview specifically focused on the efficient training of Transformers models. Consequently, through the examination of several current efforts, and in order to give a more comprehensive overview of efficiency methodologies in ViT, we observed the necessity of a survey entirely focused on the efficiency-architectural domain. Precisely, as briefly mentioned also in [19], we categorize efficient deep learning algorithms for ViT structures into four categories by adopting the following reported methodology:

- *Compact architecture* (CA) - analyzes solutions specifically developed to reduce the computational cost of self-attention in order to guarantee the ViT global understanding of the input features while reducing (often by attention linearization) the computational cost of such architectures.
- *Pruning* (P) - focuses on strategies designed to reduce the number of neurons and connections of ViT models in order to maintain high accuracy while avoiding the model over parametrization and reducing the number of computed operations (multiplications).
- *Knowledge distillation* (KD) - analyzes learning strategies that aim to improve the performance of shallow (student) models by sharing and compressing the knowledge from deeper ones (teacher).
- *Quantization* (Q) - technologies that aim to reduce data type, from floating point to integer, and precision, from 32-bit to a lower bit-rate, of ViT's weights and activation functions in order to obtain lightweight and memory-efficient models.

Moreover, the main contributions of this work are summarized as follows:

- We present a comprehensive review of efficient ViT methodologies emphasizing the mathematical aspects, analyzing the proposed strategies, and comparing their performances on well-known benchmark datasets.
- We surveyed the milestones of efficient ViT strategy (up to 2023) for the four selected efficient categories (CA, P, KD, and Q).
- We introduce a novel evaluation metric named Efficient Error Rate (EER), which is able to consider all the parameters that can affect a general device at inference time, such as the number of parameters, bits, FLOPs, and model size, in order to compare the analyzed methodologies over well-known benchmark datasets fairly. We also identify the best strategy that better balances EER and estimation capabilities (accuracy).
- We finally provide some useful insight for future development and promising research directions.

The rest of the survey is organized as follows: Section 2 reviews some general and mathematical concepts of ViT and efficient strategies, Section 3 describes state-of-the-art efficient deep learning solutions proposed in recent years. Finally, Section 4 compares the estimation performances achieved by the previously introduced methods when applied to tackle different CV tasks, and Section 5 discusses ViT challenges and promising research directions.

2 Background

This section aims to define preliminary notions and mathematical formulations for the four categories of the analyzed efficient techniques in order to highlight better the novelties introduced in the state-of-the-art researches reported in Section 3. Precisely, following the four subset categorization of the survey, Section 2.1, Section 2.2, Section 2.3, and Section 2.4, respectively formalize vision transformers, pruning, knowledge distillation, and quantization strategies.

2.1 Background on Vision Transformers

The development of transformer architectures has been mainly due to their ability to capture long-range dependencies and incorporate more information with respect to fully CNN models when trained on very large datasets [60]. Those architectures are composed of two key elements: patch embedding and feature extraction module computed via cascaded self-attention blocks. Moreover, classification heads or other structures, such as decoders, are stacked on top of the encoding part in order to perform the desired task. For instance, in the case of a general classification task, the first element is used to process the input RGB image before the feature extraction takes place in order to obtain a sequence of embedded data. Subsequently, each embedded sequence is fed into a transformer encoder composed of multiple self-attention blocks in order to extract low-level features and complex relationships between different elements of each input sequence. Finally, a multi-layer perception (MLP) head composed of two feed-forward functions is used to compute the output probabilities. We report in the following paragraphs a more detailed mathematical formulation of the path embedding and self-attention mechanism.

Patch embedding: The first key element of a transformer structure is the creation of patch embedding; in order to obtain it, the input feature maps are usually processed as follows: at first, the input image $x \in R^{H \times W \times C}$ is divided into N patches. Similar to a word sequence in NLP tasks, a patch is a pixel matrix containing part of the input image, i.e., a subset of the input data. Each patch is then flattened in order to obtain a sequence of n entities and multiplied with a trainable embedding tensor which learns to linearly project each flat patch to dimension d; this results in n embedded patches of shape $1 \times d$, lets generally denote them as $N \in R^{n \times d}$. Subsequently, a trainable positional embedding is added to the sequence of projections in order to add the spatial representation of each patch within the image space, let's define the overall output embedding as z.

Self-attention: As a following step, given the patch embedding sequence z_n, the self-attention mechanism learns how to gather one token $t_i \in z_n$ with the others (t_j with $j \neq i$ and $i, j \in d$) into the sequence. This solution leads to global information extraction from the input features, which improves the fixed receptive field of well-known convolution operations. Usually, the transformer structure is based on a multi-head self-attention (MSA) mechanism, which is composed of several single self-attention layers running in parallel; we report a graphical overview of the just introduced operations in Figure 1.

Therefore, the self-attention module can be mathematically defined as follows. Given an input vector, this operation first computes three matrices: the query Q, the key K, and the value V, respectively, with equal sizes ($d_q = d_k = d_v$). Subsequently, the operation translates the obtained scores into probabilities, computing the $softmax$ function. Consequently, the Vanilla self-attention [62], also known as softmax dot-product self-attention operation, can be formally defined with the attention matrix $A \in R^{n \times d_v}$ as reported in Equation 1 where the output matrix A updates each component of a sequence by aggregating global information from the complete input sequence.

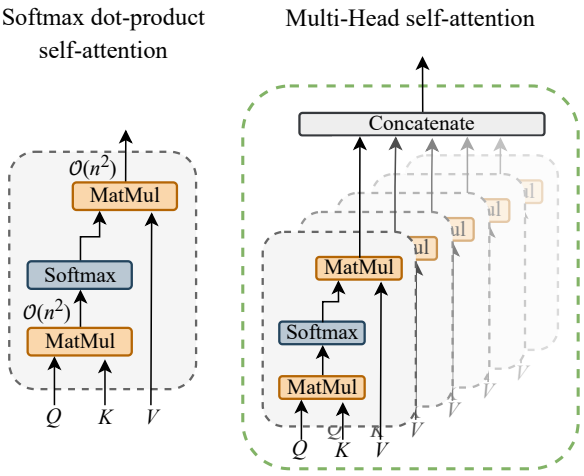

Fig. 1. Graphical overview of the vanilla self-attention and the multi-head self-attention blocks. The $\mathcal{O}(n^2)$ in the softmax dot-product self-attention highlights the quadratic cost of each operation.

$$A(Q, K, V) = Softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \tag{1}$$

However, from a computational complexity aspect, the time and memory cost for this operation increases quadratically with the number of patches n within an image, i.e., $\mathcal{O}(n^2)$. This cost is due to two dot product operations (reported as MatMul in Figure 1), i.e., the computation of $\frac{QK^T}{\sqrt{d_k}}$, which takes $\mathcal{O}(n^2 d_k)$, and the second one between the $softmax$ probability and V which takes $\mathcal{O}(n^2 d_v)$. Then, self-attention modules are concatenated together into an MSA in order to extract information from different areas at the same time. Finally, the inputs and output of the MSA module are normalized ($Norm$) and passed into a feed-forward network (FNN) consisting of two feed-forward layers interleaved by a nonlinear activation function (usually a GeLU). Then, by defining as X the input features of the transformer module, it is possible to formulate its output X_{out} as expressed in Equation 2:

$$\begin{aligned} X_{MSA} &= Norm(MSA(X, X)) + X \\ X_{out} &= Norm(FNN(X_{MSA})) + X_{MSA} \end{aligned} \tag{2}$$

Cross-attention: Based on a similar computation to the self-attention operation, the cross-attention mechanism is also widely employed in vision transformer applications after the patch embedding calculation. More in detail, the cross-attention leverages the use of two separate embedding sequences, computing the matrix Q from the first stream and K and V from the second one, i.e., the patch embedding operation is performed twice for each embedded data. This solution allows the conditioning of deep learning models with extra information, such as text prompts or other images.

2.2 Background on Pruning

The neural network pruning strategy, introduced by Janowsky et al. [27] (1989) and Karnin et al. [29] (1990),

is inspired by the synaptic pruning in the human brain. During this procedure, only a subset of the connections, i.e., axons and dendrites, will remain connected between the childhood and puberty phases. In neural networks, this solution does not require interfering with training procedure and loss functions; precisely, the pruning strategy is computed after the training phase in order to reduce the number of neurons and connections of the network; in fact, it is also known as post-training pruning. This procedure is performed by setting to zero some neuron's weights, usually, the ones with the lower estimation contribute (saliency), i.e., neurons that have minimal impact in the final prediction. Generally speaking, this strategy will result in lighter models with respect to the original ones, which are usually called "sparse" models due to the spare (i.e., with zeros) matrix obtained after pruning. More in detail, under the assumption that the used framework is able to take advantage of sparse computation on CPU/GPU devices, this strategy will reduce the number of multiplications between layers at inference time, saving hardware computation and improving the model's inference frequency. Moreover, by defining with D the dataset, $f(x, W)$ the trained model, where x is the input data ($x \in D$), and W its weights, we can formalize the pruning algorithm as reported in Algorithm 1.

Algorithm 1 Pruning strategy

Input: $f(\cdot,\cdot)$, trainable neural newton
D, set training of data
$P_{0/1}$, matrix of pruned features

1: $f(\cdot, W_0) \leftarrow weights\ initialization$
2: $f(x \in D, W') \leftarrow train\ until\ convergence$
3: $f(x \in d \subseteq D, P_{0/1} \odot W') \leftarrow pruning$

Precisely, after the training phase, the pruning strategy is obtained by multiplying the trained weights W' by a diagonal binary matrix $P_{0/1} \in \{0,1\}^{|W'|}$ which is composed of a set of pruned features $p_{0/1}$ with $p_{0/1} \in \{0,1\}^{\mathbb{R}}$. Moreover, to select the latter features it is commonly used a subset $d \subseteq D$ in order to compute the weights' importance score. Consequently, after the pruning operation, only a subset of the trained weights $W_{P_{0/1}} \in W'$ will not be set to zero.

However, due to the reduction of the number of neurons and connections inside the original model, this strategy usually leads to a reduction of the generalization capabilities with respect to the original model. Therefore, in order to define the best trade-off between zeroed weights (fast inference) and estimation performances, the developed pruning strategies will mainly differ on how to identify the weights that can be zeroed with the minor accuracy reduction, i.e., in how to define the $p_{0/1}-$vector optimally. In order to give a general overview, pruning techniques usually focus on three sets of studies: (1) *structure* and *unstructured* methods, i.e., strategies that focus on the entire set of weights or on specific ones. (2) *score computation*, which are the possible ways to compute the pruning vector and the identification of the number of network's weights that can be pruned. (3) *training phase*, which focuses on the training-pruning (or fine-tuning) strategy employed at the training phase.

2.3 Background on Knowledge Distillation

The knowledge distillation (KD) strategy has been introduced by Hilton et al. [22] (2015) in order to reduce the computational requirements of deep neural networks with respect to other solutions like ensemble learning and the mixture of experts. This choice is particularly due to the difficult deployment of large deep neural network models on edge devices with limited memory and computational capacity. The basic idea behind this KD strategy is to transfer the generalization and estimation abilities from a deep (pretrained) model to a shallower and lightweight one through class probabilities. As commonly defined, we identify the deeper model as the *teacher* and the shallower as the *student*. Therefore, the objective of the learning strategy is to train the student model in order to match the class probabilities (p_i) produced by the teacher; the authors define these values as soft targets. Consequently, we can define the distillation loss ($\mathcal{L}_{Dstl}$) as reported in Equation 3 where p_i^t and p_i^s are respectively the soft labels of the teacher and the student and CE the cross-entropy loss function.

$$\mathcal{L}_{Dstl} = CE(p_i^t, p_i^s) \tag{3}$$

The described procedure allows the smaller model to minimize the distance from the teacher's output distribution, i.e., learning from information that is not provided by the ground-truth labels, and consequently, closely mimic the behavior of the pretrained large teacher. Moreover, by adding to the overall optimization problem the cross-entropy loss computed between the hard labels (classification vector) of the student and the ground truth labels, which we define as ($\mathcal{L}_{Class}$), we can formulate the loss function of the vanilla KD learning strategy ($\mathcal{L}_{KD}$) as reported in Equation 4.

$$\mathcal{L}_{KD} = \mathcal{L}_{Class} + \mathcal{L}_{Dstl} \tag{4}$$

Furthermore, to give a better understanding of the newly introduced learning strategy, we show in Figure 2 its block diagram representation.

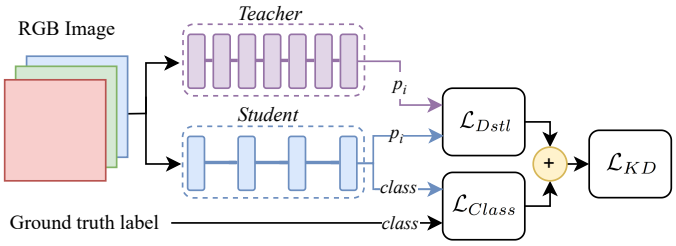

Fig. 2. Graphical representation of the vanilla KD learning strategy. Please refer to Section 2.3 for the used notation.

2.4 Background on Quantization

The last compression approach that we review in this work is the quantization procedure. The quantization has the objective of reducing the neural network parameters (weight and activation values) from floating point precision data types, i.e., usually 32-bit, to a lower bit representation,

TABLE 1
Summary of all the analyzed efficient vision transformer models. We organize the modes following their categorization, i.e., compact architecture design (CA), pruning methods (P), knowledge distillation (KD), and quantization (Q). We also underline methods that combine (+) different strategies.

Category	Model / Paper	General overview	Code
CA	PVT (Wang et al., 2021)	Propose the SRA attention module which has a computational cost R_i^2 time lower than MSA	✓
	Swin Transformer (Liu et al., 2021)	Propose a shifted window attention (non overlapping patches) against vanilla sliding window	✓
	SOFT (Lu et al., 2021)	Introduce a softmax-free attention module obtained via low-rank decomposition strategy	✓
	PoolFormer (Yu et al., 2022)	Replace the attention with a non-parametric pooling operation to reduce the computation	✓
	PVTv2 (Wang et al., 2022)	Introduce a Linear SRA attention module which leverage the pooling spatial reduction capabilities	✓
	MViTv2 (Li et al., 2022)	Limited computational complexity with residual pooled attention	✓
	SimA (Koohpayegani and Pirsiavash, 2022)	In order to avoid $exp(\cdot)$ operations, authors propose a softmax-free attention solution	✓
	Flowformer (Huang et al., 2022)	Introduce an attention module inspired by flow neural networks	✓
	Hydra Attention (Bolya et al., 2022)	Linear attention based on decomposable kernel strategies and an high amount of attention heads	✓
	Ortho (Huang et al., 2022)	Reduce the computational complexity of MSA orthogonalizing the tokens within local regions	✗
	Castling-ViT (You et al., 2023)	Propose a fully-linear ViT architecture thought a linear-angular attention module	✓
	EfficientViT (Cai et al., 2023)	Propose a fully-linear ViT architecture designed for high-resolution dense prediction tasks	✓
CA + Q	EcoFormer (Liu et al., 2023)	Propose to kernelizing the attention module via hash functions to reduce its computational cost	✓
P	VTP (Zhu et al., 2021)	Propose a strategy to prune attention modules	✗
	DPS-ViT (Tang et al., 2022)	Propose a path slimming algorithm to identify and discard redundant patches	✗
	WDPruning (Yu et al., 2022)	Introduce a solution to reduce both weight and depth of a ViT	✗
	NViT (Yang et al., 2023)	Propose an Hessian-based global structured pruning algorithm	✓
	X-Pruner (Yu and Xiang, 2023)	Design a layer-wise pruning algorithm based on eXplainable AI masks	✓
P + KD	DynamicViT (Rao et al., 2023)	Leverage to progressively and dynamically prune less informative ViT tokens	✓
KD	DeiT (Touvron et al. 2021)	Introduce a distillation token into the attention module to mimic teacher's hard labels	✓
	Monifold Distillation (Hao et al., 2022)	Introduce a loss function based on patch-level information present in transformers modules	✗
	TinyViT (Wu et al., 2022)	Introduce a new family of small ViT which is trained via memory efficient KD strategy	✓
	DearKD (Chen et al., 2022)	Introduce a distillation framework for limited/data-free training	✗
	CivT (Ren et al., 2022)	Distillation strategy based on multiple teachers which extract features from different perspectives	✓
	MiniViT (Zhang et al., 2022)	Introduce a weights multiplexing strategy applied to ViT architectures	✓
	SMKD (Lin et al., 2023)	Propose a patch-masking approach for ViT applied to few-shot learning tasks	✓
Q	- (Liu et al., 2021)	Introduce a mixed-precision weights strategy formulated as an optimization problem	✗
	PTQ4ViT (Yuan et al., 2022)	Twin uniform strategy, which splits negative and positive weight's values into two bit-ranges	✓
	APQ-ViT (Ding et al., 2022)	Popose a Matthew-effect preserving scheme for ultra-low bit quantization	✗
	Auto-ViT-Acc (Lit et al., 2022)	Introduce a ViT quantization strategy specifically designed for FPGA devices	✗
	NoisyQuant (Liu et al., 2023)	Improve previous quantization methods by adding a fixed noisy factor to the date-distribution	✗
Q + P + KD	GPU-SQ-ViT (Yu et al., 2023)	Introduce a GPU friendly framework based on a mixed KD strategy	✗

such as 8/6/4/2-bit precision and/or to a different data type, i.e., integers. Thus, quantized models will significantly save storage memory and speed up the inference process. However, due to the strong compression, the model could suffer severe accuracy drops or even instabilities.

The quantization function $\Psi(\cdot,\cdot)$ and the quantization

intervals Δ (also known as scaling factors, i.e., the way to represent the data with the available bit-width) are the two main critical features of this approach, which should be carefully chosen. To give a general overview, by defying with x a floating-point value and with k the number of available bits, we report in Equation 5 the most popular function, namely the uniform quantization function, where the data range is equally split.

$$\Psi_k(x, \Delta) = Clamp\left(Round\left(\frac{x}{\Delta}\right), -2^{k-1}, 2^{k-1} - 1\right) \quad (5)$$

Moreover, a graphical representation of the quantization procedure from the commonly used 32-bit representation to a general *k*-bit compression is reported in Figure 3.

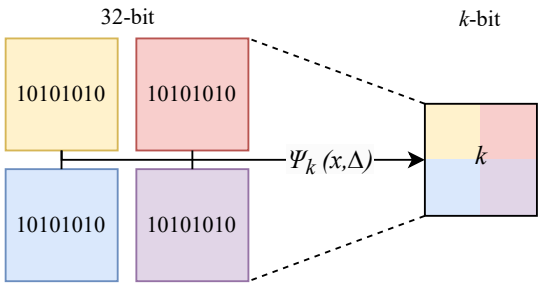

Fig. 3. Graphical representation of a general quantization procedure; the floating-point 32-bit data is compressed based on the quantization function $\Psi_k(x, \Delta)$ to a *k*-bit representation.

Precisely, there exist two types of quantization methods: the quantization-aware training (QAT) and the post-training quantization (PTQ). The first strategy interleaves training and quantization procedures, i.e., finetuning the quantized model at low precision, while the second is computed after the training phase without the need for re-training or fine-tuning. However, for ViT architecture, the QAT strategy is highly costly due to the re-training phases; differently, PTQ could be a preferable solution, enabling a fast quantization and deployment, needing a few samples (also unlabelled) to calibrate the quantization procedure.

3 Efficient Vision Transformer

This section reviews architectures and strategies proposed in recent years in order to design efficient ViT models. Precisely, following the four subset categorization of the survey, Section 3.1, Section 3.2, Section 3.3, and Section 3.4, respectively analyzes the compact ViT, pruning, knowledge distillation and quantization strategies specifically designed for ViT models. We report in Figure 4 a graphical overview of the compression technique behaviors and a summary of the reviewed models in Table 1.

3.1 Compact architecture design

Based on the mathematical basis introduced in Section 2.1, in this paragraph, we review multiple researches focused on architectural-like optimizations, i.e., solutions that are designed to reduce the computational cost of ViT by reducing or linearizing the cost of the self-attention module. In order to give a better understanding of how these technologies have developed, the proposed solutions are reviewed to address the task according to their release date. Such a straightforward solution will emphasize a gradual reduction of the self-attention's computational cost. In particular, we will point out that earlier research (2021) will reduce the quadratic computational cost without attaining a fully-linear design, as will be proposed in more recent studies (2023).

A preliminary work has been conducted by Wang et al. [65] (2021), who propose Pyramid Vision Transformer (PVT), a versatile convolution-free (pyramidal) backbone for dense prediction tasks. The ViT architecture uses a progressive shrinking strategy based on a pyramidal structure in order to handle high-resolution feature maps and reduce their computational cost. The paper introduces and substitutes the standard multi-head attention (MSA) with spatial-reduction attention (SRA), lowering the computational/memory complexity of attention operation. Moreover, the SRA reduces the resource consumption of ViTs while making the PVT model flexible to learning multi-scale and high-resolution features, i.e., working with high-resolution images. Then, the SRA is R_i^{21} times lower than the standard MSA; therefore, the model can handle high-resolution input features with lower computational/memory requirements. Furthermore, by defining with $SR(\cdot)$ the operation for reducing the spatial dimension of the input sequence, it is possible to formally define the single attention module A_{SRA} of the SRA[2] in Equation 6.

$$A_{SRA}(Q, K, V) = A(Q, SR(K), SR(V)) \quad (6)$$

During the same year, Liu et al. [46] (2021) propose Swin Transformer, an architecture able to achieve a linear computation complexity to input image size. This result is achieved by computing a local multi-head self-attention operation, i.e., only within each (non-overlapping) window (W-MSA). This choice avoids a global self-attention operation as proposed in Dosovitskiy et al., taking advantage of smaller feature windows obtained aggregating neighboring patches of the input feature maps. Moreover, the authors introduce a *shifted window* (SW) operation computed between consecutive self-attention layers. In regards to the standard *sliding window* operation, the proposed solution, combined with the self-attention mechanism, namely shifted W-MSA (SW-MSA), leads to notable latency improvements in real-world applications facilitating memory access in hardware.

Differently from previous studies, Lu et al. [49] (2021) propose SOFT, a softmax-free transformer model. The authors identify the high computational cost of vision transformer architectures in the softmax operation (i.e., $exp(\cdot)$) computed in the self-attention layer, therefore proposing to approximate the operation via low-rank decomposition. Therefore, the final formulation of the softmax-free self-attention (A_{SOFT}) is formulated as follows, the authors generate $\tilde{Q}$ and $\tilde{K}$ with a convolution and average pooling operations from the query Q and key K respectively.

$$A_{SOFT} = exp(Q \ominus \tilde{K}) \cdot \left(exp(\tilde{Q} \ominus \tilde{K})\right)^{\dagger} \cdot exp(\tilde{Q} \ominus K) \quad (7)$$

1. R_i is the reduction ratio of the attention layers in Stage i.
2. SRA is then obtained as MSA concatenating h attention modules

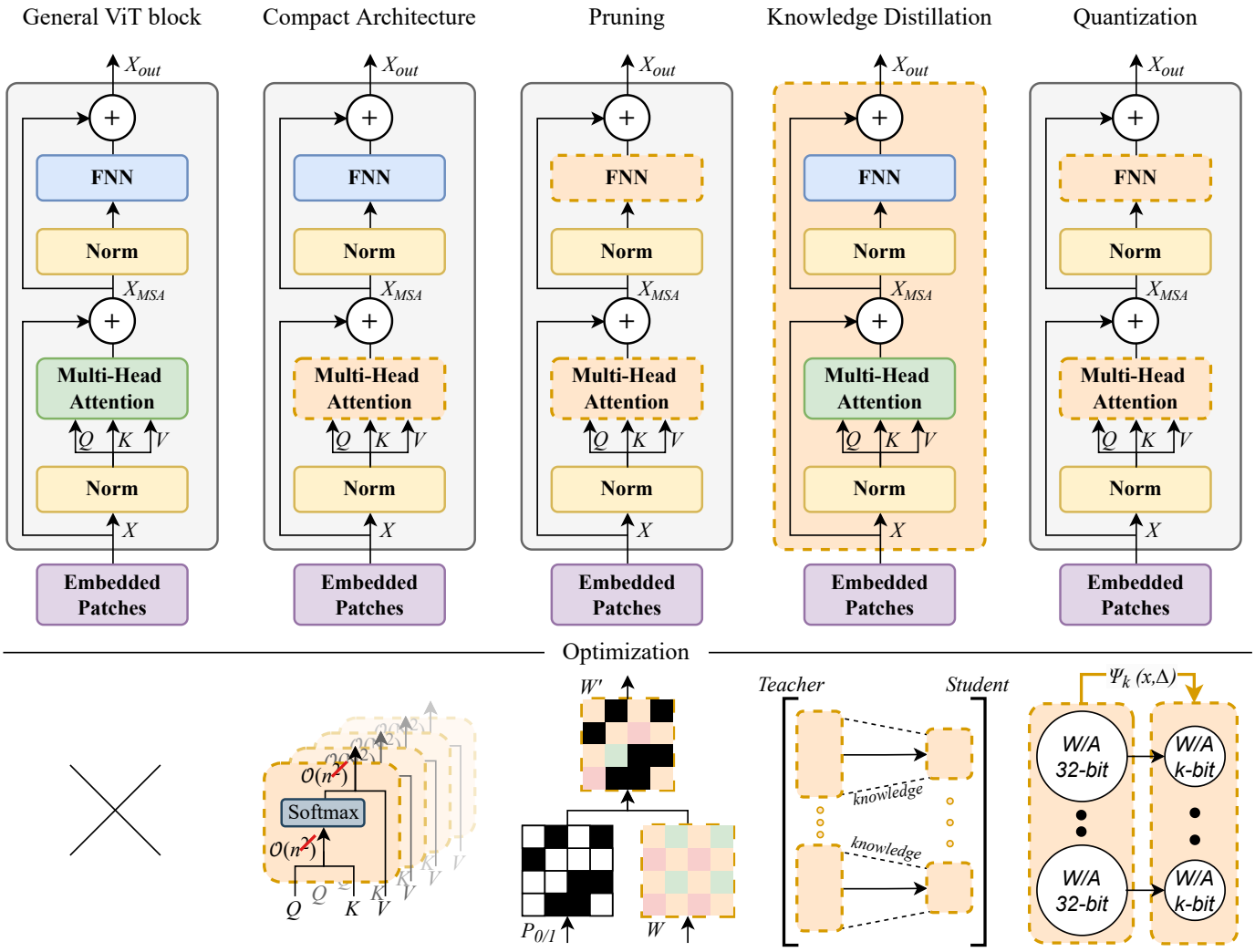

Fig. 4. Graphical representation of efficient ViT techniques and their optimization effects. The fundamental element on which the VITs are built is shown on top, where the dashed orange block highlights the component on which each optimization technique mainly focuses. A graphical depiction of how the optimizations influence the block of interest is also provided at the bottom of the image. Please refer to Section 2 for the reported variables and their description.

The following year, Yu et al. [77] (2022) propose a study on the impact of the token mixer in vision transformers; the authors abstract the general structure into the so-called MetaFormer structure. Furthermore, by replacing commonly used token mixers, such as the self-attention operation, with a non-parametric pooling ($Pool$) operator, Yu et al. designed PoolFormer. Therefore, given the input features X, the MSA module of PoolFormer ($X_{PF_{MSA}}$) can by formulated as reported in the following equation.

$$X_{PF_{MSA}} = Pool(Norm(X) + X) \tag{8}$$

Based on the reported formulation, thanks to the pooling operation, the proposed model is able to achieve a linear computational complexity to the token's sequence length without the addition of any learnable parameter.

Li et al. [36] (2022) propose MViTv2, an improvement of the previously proposed MViT backbone by Haoqi Fan et al. [17] (2021). More in detail, MViTv2 improves the pooling self-attention introduced in MViT [17], with a residual pooling connection, in order to keep a limited computational complexity and reduced memory requirements of the attention block while increasing the information flow and facilitate the training process. This operation is performed by adding a pooled query tensor $Pool(Q)$ to the output sequence inside the MViT attention module. Therefore, by defining the pooling self-attention A_{MViT} as reported in Equation 9, its improved version proposed A_{MViTv2} can be formulated as reported in Equation 10.

$$A_{MViT} = A\big(Pool(Q), Pool(K), Pool(V)\big) \tag{9}$$

$$A_{MViTv2} = A_{MViT} + Pool(Q) \tag{10}$$

Moreover, inspired by prior works that employ the pooling operation in order to reduce the computational cost of self-attention, Wang et al. [64] (2022) provide an enhanced version of PVT, namely PVTv2. The innovative architecture is designed on a novel linear spatial-reduction attention (Linear SRA) module, which uses an average pooling to

reduce the spatial dimension before the attention operation rather than the convolutional one utilized in the previous SRA module. In addition, the authors also introduce an overlapping patch embedding to prevent losing some of the image's local continuity.

Motivated by the issue highlighted by Lu et al. [49] in 2021, Koohpayegani and Pirsiavash [34] (2022) provide an enhanced strategy to manage the exponential operation computed by the softmax layer. The authors introduce SimA, a softmax-free attention block, in which the softmax layer is replaced with a l_1-normalization operation in order to normalize the query ($\hat{Q} = ||Q||_1$) and key ($\hat{K} = ||K||_1$) matrices. This solution modifies the computation of vanilla self-attention into a standard sequence of multiplications between matrices. More in detail, similarly to Lu et al. [49], this choice is due to the $exp(\cdot)$ operation computed in the softmax layer, which, in a transformer architecture, does most of the computation when the input feature map is large. However, differently from Lu et al. [49], SimA is not based on low-rank decomposition; instead, it is based on an adaptation method in order to linearize the transformer's computational cost at test time. Moreover, the authors demonstrate the effectiveness of the proposed solution applied to state-of-the-art models such as DeiT, XCiT, and CvT, achieving on-par accuracy with respect to standard self-attention layers. Although the overall computational complexity has been reduced (it is more efficient when $n >> d$), the overall cost of the proposed model is still quadratic with respect to the number of input tokens ($\mathcal{O}(n^2d)$) or the dimension of each token ($\mathcal{O}(nd^2)$). However, SimA is able to dynamically choose the smallest computational complexity based on the number of input tokens linearizing the overall computational cost ($\mathcal{O}(nd)$); the SimA attention module A_{SimA} can be formulated as reported in Equation 11.

$$A_{SimA} = \begin{cases} \hat{Q}(\hat{K^T}V) & \text{if } n > d \\ (\hat{Q}\hat{K^T})V & \text{otherwise} \end{cases} \tag{11}$$

Summarizing, SimA is able to adapt its computation at the test time in order to achieve a linear computation on the number of tokens or on the number of channels while being more efficient on edge and low-power devices thanks to the softmax-free solution, i.e., lack of $exp(.)$ operations.

Differently from all the previous studies, Huang et al. [26] (2022) propose Flowformer, a transformer architecture based on *Flow-attention* modules. The optimized attention layer is inspired by flow networks, architectures defined as a directed graph where each edge has a capacity and receives a flow of information, as described by Waissi et al. [63] (1994). Precisely by non-negative and non-linear projection ϕ of flow computation capacity. The introduced operation aggregates the weights (i.e., attention maps) computed from the queries (Q) and keys (K) over the information values (V), achieving a linear complexity with respect to the number of input tokens. The authors define the capacity of conserved information flows as $\hat{I} \in \mathbb{R}^{n\times 1}$, the incoming flow, and with $\hat{O} \in \mathbb{R}^{m\times 1}$ the outgoing flow. Therefore, the overall flow attention module A_{Flow} can be formulated based on flow conservation principles [63] (competition, aggregation and allocation) as reported in Equation 12.

$$\begin{aligned} &Competition : \hat{V} \in \mathbb{R}^{m\times d} = Softmax(\hat{O}) \odot V \\ &Aggregation : A \in \mathbb{R}^{m\times d} = \frac{\phi(Q)}{I} \cdot \left(\phi(K)^T \cdot \hat{V}\right) \\ &Allocation : A_{Flow} \in \mathbb{R}^{n\times d} = Sigmoid(\hat{I} \odot A) \end{aligned} \tag{12}$$

Furthermore, Bolya et al. [3] (2022), inspired by the use of decomposable kernel strategies [30] (linear in the number of tokens) in order to reduce the computational cost of self-attention, proposed to take a step further by developing the Hydra Attention module. The proposed technique is based on the concept that increasing the number of attention heads in the MSA does not increase the computational cost. Consequently, aligning the number of attention heads with the features in a decomposable kernel strategy, the model achieves computational linearity in both tokens and features, resulting in a reduced computational cost of the attention layer. Moreover, as reported in the original paper, this method is particularly effective for high-resolution images, a common bottleneck for high-demanding architectures.

Huang et al. [25] (2022) propose a general-purpose backbone called Orthogonal Transformer (Ortho), which is able to reduce the computational cost of standard attention mechanism with an orthogonal self-attention (A_{OSA}). The attention is computed orthogonalizing the tokens within local regions, permuting them into token groups; the orthogonal matrix is defined as $O \in \mathbb{R}^{n\times n}$. Finally, the MSA is computed group-wisely into the orthogonal space. This solution leads to the possibility of computing the A_{OSA} with a lower resolution with respect to the commonly used image space, reducing the overall computational complexity of self-attention of a factor n_0 equal to the number of groups into which the tokens are separated. The authors also introduce a Positional MLP in order to incorporate position information for arbitrary input resolutions into Ortho. Therefore, the output of the attention module can formalized as reported in Equation 13; the authors employ the inverse orthogonal matrix after the MSA in order to recover the visual tokens from the orthogonal representations.

$$X_{out} = O^T \cdot X_{MSA}(O) \tag{13}$$

Recently, Liu et al. [42] (2023) propose to reduce the computational cost of self-attention by taking advantage of an extreme quantization technique. Consequently, we characterize this study as CA + Q since, similar to previous models, as it aims to reduce the computing cost of the self-attention module by leveraging the use of a quantization (Q) methodology. The authors introduce EcoFormer, a binarized self-attention module that extensively reduces the multiply and accumulates (MACs) operations in standard attention layers in order to save a considerable on-chip energy footprint. More in detail, by defining with D_p the number of dimensions for each attention head and b the number of bits, the proposed solution, based on kernelized hash functions H, with $H \in \mathbb{R}^{D_p} \mapsto \{1, -1\}^b$, is particularly effective for edge devices, where the computational resources are a bottleneck for high capacity deep learning architectures. Therefore, inspired by the idea of kernel-based linear atten-

tion [52], the attention equation (A_{Eco}) can be formalized as follows.

$$A_{Eco} = Softmax\left(\frac{H(Q)^T \cdot H(K)}{\sqrt{d_k}}\right) \cdot V \tag{14}$$

The authors take advantage of an *extreme* quantization scheme ($b = 16$ hashing bit), which is able to represent feature vectors[3] in binary codes. However, although the solution is demonstrated to be energy and memory saving, the binary compression requires specialized GPU kernels to be piratically deployed on edge devices; therefore, these limitations result in a bottleneck on the effective efficiency of EcoFormer self-attention modules when employed in real-case scenarios.

Finally, You et al. [73] (2023) introduced a novel framework named Castling-ViT, which aims to be a ViT structure composed of only linear terms. To tackle this problem, the authors introduce a linear-angular attention module, where angular kernels are decomposed in linear terms, and the remaining high-order residuals are approximated with a depth-wise convolution and an auxiliary masked-self-attention operation, i.e., an attention module which only focuses on a limited number of patches. Although the latter element still has a quadratic computing cost, the authors noted that it tends to converge to zero during the training phase, thus being worthless during the inference phase. Moreover, by defining the angular kernel ($Sim(Q, K)$) as a similarity measurement function computed between the queries Q and keys K, the linear-angular attention module (A_{Cast}) can be formulated as reported in Equation 15.

$$A_{Cast} = Sim(Q, K) \cdot V \tag{15}$$

In contrast to previous studies focused on efficient backbones for general-purpose applications, Cai et al. [5] (2023) propose EfficientViT, an efficient ViT architecture with linear computational cost designed to handle high-resolution dense prediction tasks such as semantic segmentation and super-resolution. The authors leverage the use of the ReLU-based global attention [30] to achieve both the global receptive field of ViT and linear computational complexity.

3.2 Pruning

This section reviews the state-of-the-art ViT pruning method based on the background notions and mathematical formulations introduced in Section 2.2. Similar to previous analysis, we review the proposed solutions to prune ViT architectures according to their release date. In particular, we will emphasize how pruning algorithms evolved in recent years over both strategies to compute the importance score and adaptive solutions for the pruning/preserving ratio in order to maximize the performance of the algorithm.

A preliminary study has been performed by Zhu et al. [84] (2021), who propose VTP, a vision transformer pruning method that is able to thin out ViT architectures while encouraging dimension-wise sparsity. The solution mainly focuses on MSA and FNN transformer structures in order to identify less informative features, i.e., by reducing the number of embedding/neuron dimensions via control coefficients. As a common pruning strategy, VTP lays the groundwork on a feature importance score (p). This strategy is based on learning at training time the soft pruned features $\hat{p}_{0/1} \in \{0, 1\}^{\mathbb{R}}$, while defining the hard pruned features $p^*_{0/1} \in \{0, 1\}^{\mathbb{N}}$ at inference time, based on a threshold value $\tau \in \{0, 1\}^{\mathbb{R}}$, i.e., $p^*_{0/1} = \hat{p}_{0/1} \geq \tau$. Then, the hard pruned self-attention transformer block (A^*_{VTP}) can be formulated as reported in Equation 16, where $P_{0/1}$ is a diagonal matrix whose diagonal line is composed of $p^*_{0/1}$-elements, i.e., $P_{0/1} = diag(p^*_{0/1})$.

$$A^*_{VTP}(Q^*, K^*, V^*) = P_{0/1} \cdot A(Q, K, V) \tag{16}$$

Precisely, if the algorithm attributes a value of $p = 0$ to the feature, it will be discarded; otherwise ($p = 1$), it will be maintained.

Moreover, Tang et al. [57] (2022), inspired by a previous approach on CNN architectures proposed in [48], introduce PS-ViT, a patch slimming framework to improve the efficiency of vision transformers structures. The proposed solution is motivated by the common problem of overparameterization and redundant information of deep learning algorithms. Therefore, PS-ViT mainly focuses on identifying transformer patches with redundant information in order to discard them and accelerate the inference process. As common pruning algorithms, in PS-ViT, each patch in the attention bock receives an importance score $p = \{0, 1\}^{\mathbb{N}}$. Differently from VTP, the $p_{0/1}$-values are learned during the backpropagation phase with a top-down procedure, i.e., from from the output layer to the input one. Consequently, the pruned attention module A^*_{PS-ViT} can be reformulated as reported in Equation 17, where $P_{0/1}$ is a diagonal matrix whose diagonal line is composed of $p_{0/1}$-elements.

$$A^*_{PS-ViT}(Q^*, K^*, V^*) = P_{0/1} \cdot A(Q, K, V) \tag{17}$$

Precisely, if the algorithm attributes a value of $p = 0$ to the patch, it will be identified as redundant and will be discarded; otherwise ($p = 1$), it will be maintained. Moreover, Tang et al. propose a dynamic variant of the patch slimming algorithm, named DPS-ViT, which adaptively selects the non-redundant features at inference time depending on the input samples.

Furthermore, Yu et al. [75] (2022), motivated by the fact that previous pruning strategies only focus on architecture width, develop the Width & Depth Pruning (WDPruning) framework. The proposed solution reduces the architecture width via a learnable saliency score threshold, similar to previous works, while limiting the structure depth by introducing multiple classifiers inside the model. WDPruning framework has the final objective of determining an optimal trade-off between accuracy and efficiency, with a shallower model from both depth and width perspectives. Precisely, the width pruning is commonly based on a diagonal binary matrix $P_{0/1}$, while the threshold (τ) dynamically updates the pruning ratio via the Augmented Lagrangian method at the training phase until reaching a predefined score. Moreover, the depth pruning procedure has the objective of identifying the shallower classifier; this procedure is computed via several classifiers plugged into the architecture, which are evaluated at validation time in order to determine

3. Weights and operations in Deep Learning models are usually computed at floating point precision data type.

an optimal trade-off between estimation performances and efficiency.

Based on a different approach with respect to previous works, Yang et al. [70] (2023) propose a global[4] structural pruning criteria, which is able to guarantee parameter redistribution along the network. The authors propose to compute the importance score (p) as the Hessian matrix norm of the loss, which, in contrast to local pruning strategies focusing on specific layers/neurons, leads to a global pruning of the overall architecture. Moreover, this solution, applied to the DeiT-Base model, enables the generation of new efficient ViT models named NViT.

Recently, Rao et al. [53] (2023), extend the previously developed DynamicViT framework [54] (2021), utilized to increase sparsity and accelerate the inference of general neural network structures over ViT architectures. The framework bases the groundwork on [57], where only a subset of image patches are needed for the final prediction. Therefore, in DynamicViT, the authors propose to progressively and dynamically[5] prune less informative tokens hierarchically based on a prediction module. The latter structure is placed between Transformer blocks in order to individuate and discard less informative tokens, while a threshold value is used as a ratio to determine the percentage of the information to be preserved. Moreover, in order to minimize the influence on performance drop caused by spatial sparsification, the authors also leverage knowledge distillation techniques in order to guide the pruned-student model to the teacher behavior. Precisely, the original backbone network is used as the teacher model, while an equal structure is dynamically pruned in order to obtain the student variant. In addition, motivated by DynamicViT, researchers investigate methods for handling adaptive inference approaches in ViT structures based on pruning/merging of less informative tokens; we provide an in-depth study of such methodologies, such as [71, 33, 4], in the supplementary material Section 6.1.

Finally, Yu and Xiang [76] (2023) proposed X-Pruner, a layer-wise pruning algorithm that leverages eXplainable AI (XAI) principles in the pruning strategy. The baseline idea is to identify and prune less contributing attention units from an explainability perspective. Specifically, the authors introduce a learnable explainability-aware mask (M) that can be used to prune or unprune the model based on an adaptive threshold value (θ). Moreover, by defining the threshold ratio r, two hyperparameter values respectively set to $h_1 = 10$ and $h_2 = 500$, and with Φ a function that returns the top $(1-r)\%$ sorted elements of M, the desired mask $\hat{M}$ for a generic layer can be formally defined as reported in Equation 18.

$$\hat{M} = \begin{cases} Mtanh(h_1(M-\theta)) & \text{if } M \in \Phi(M|1-r) \\ h_2 tanh(h_1(M-\theta)) & \text{otherwise} \end{cases} \quad (18)$$

4. Differently from local pruning, the global one focuses on all the network's parameters in order to prune a fraction of them.

5. In a dynamic pruning framework, during the training phase, the model learns how to select the most/least informative tokens (based on a score) so that at the inference phase, the model is able to classify them according to the specific input features, thus reducing the computation and increasing the model's throughput. In contrast, the pruning percentage of the reference model is usually chosen a priori.

3.3 Knowledge Distillation

This section reviews knowledge distillation learning techniques specifically designed to prove and design lightweight ViT models. Please refer to Section 2.3 for background information. Similar to previous analysis, we review KD solutions for ViT architectures according to their release date. In particular, we will emphasize how the well-known vanilla KD strategies have been first adapted to ViT architectures and successively improved in order to maximize the student's ability to mimic the teacher's behavior.

Touvron et al. [61] (2021) have been the first to explore the KD learning strategy for ViT architectures by introducing a token-based strategy denoted by DeiT. The proposed transformer-specific approach is based on distillation through attention, i.e., the authors add a distillation token into the self-attention module, which aims to reproduce the class (hard) label estimated by the teacher. Therefore, the token will interact with both student attention and layers and teacher labels learning the hard labels during the backpropagation.

Moreover, motivated by the high computational cost of ViT and their difficult application on edge and low-power devices, Hao et al. [20] (2022) explore the patch-level information present in transformers modules in order to propose a fine-grained manifold distillation method. This manifold strategy learns a smooth manifold ($\mathcal{M}$) embedded in the original feature space to construct low-dimensional features ($\psi(F)$). Precisely, the authors train the student model to match a pretrained teacher in the patch-level manifold space. The proposed solution led to the introduction of the manifold distillation loss function ($\mathcal{L}_{mf}$), reported in Equation 19.

$$\mathcal{L}_{mf} = ||\mathcal{M}(\psi(F_S)) - \mathcal{M}(\psi(F_T))||_F^2 \quad (19)$$

Wu et al. [68] (2022) propose a new family of architectures named TinyViT. The structure of these models is obtained via a constrained local search [24] algorithm in the model space spanned by multiple constrained factors and is trained with a memory-efficient KD strategy. Precisely, the latter procedure focuses on storing sparse[6] soft-labels of deep and heavy pretrained models into storage devices. Consequently, at training time, it will be possible to reuse the stored information in order to replicate the vanilla KD procedure while omitting the forward computation and memory occupation of the large teacher model. Moreover, the search algorithm used to generate the TinyViT architecture's family is constrained to computationally demanding ViT elements such as the depth of the model and patch size.

Differently to previous methods that apply the KD strategy from a ViT teacher to a ViT student model, Chen et al. [8] (2022), motivated by the high number of real samples needed to train ViT architectures, propose a two-stage learning framework, named Data-efficient EARly Knowledge Distillation (DearKD). The first stage is composed of a vanilla KD learning strategy where, inspired by [11], CNN features extracted from both the inner and classification layers of the models are distilled with transformer tokens. Moreover, in the second stage, the ViT student model is trained without distillation. However, in the case of a

6. Selecting only the top-K soft values from the classification vector.

limited number (or data-free) of real samples, during this second phase, the authors introduce a boundary-preserving intra-divergence loss based on DeepInversion [72], which helps the learning procedure to keep the easiest positive samples away from others in the latent space without changing its boundaries.

Similar to Chen et al., Ren et al. [55] (2022), propose a KD strategy that is not based on distilling the knowledge from teacher and student models with similar architecture structures, i.e., two ViT, but via different structures. Precisely, Ren et al. rely on the idea that teacher models with different inductive biases could extract different features, i.e., looking at the samples from different perspectives and consequently co-advising the student transformer in order to improve its estimation performances. Therefore, the proposed strategy is based on vanilla KD procedure applied to two lightweight teachers, a CNN and an involution neural network (INN)[7], which focuses on different input samples' features, despite that they are trained on the same dataset, leading to an improvement of the ViT student (CivT) accuracy at training phase.

Zhang et al. [81] (2022) propose MiniViT, a compression strategy based on KD to generate *Mini*-versions of well-known ViT models, such as Mini-Swins and Mini-DeiTs respectively, from the original (teacher) Swin and DeiT transformers architectures, The proposed strategy lays the groundwork on the weight multiplexing process, which is applied to both attention matrices and feed-forward networks. More in detail, the process focuses on weight sharing, transformation, and distillation from the teacher model (t) to the student (s) one in order to improve training stability and model performance. The described procedure is applied over both attention matrices and feed-forward networks. However, due to the dimensionality reduction of the student model with respect to the teacher one, the authors apply cross-entropy (CE) losses on the relations among queries (Q), keys (K), and values (V) of the MSA by defining the self-attention distillation loss $\mathcal{L}_{att}$ and to transformer hidden states by defining the hidden-state distillation loss as $\mathcal{L}_{hddn}$ as respectively reported in Equation 21 and Equation 22. Precisely, by denoting with M_1, M_2, and M_3 the Q, K, and V matrices with equal sizes ($d_q = d_k = d_v$), with N the number of patches, and X_{out} the output features of the feed-forward network as described in Section 2.1, it is possible to compute the two previous equations as following reported.

$$R_{i,j\in\{1,2,3\}} = Softmax\left(\frac{M_i \cdot M_j^T}{\sqrt{d_k}}\right) \tag{20}$$

$$\mathcal{L}_{att} = \frac{1}{9N} \cdot \sum_{p=1}^{N} \sum_{i,j\in\{1,2,3\}} CE(R^s_{ij,p}, R^t_{ij,p}) \tag{21}$$

$$\mathcal{L}_{hddn} = \frac{1}{N} \sum_{p=1}^{N} CE(R^s_{X_{out},p}, R^t_{X_{out},p}) \tag{22}$$

7. INN, introduced in [35], has the ability to relate long-range spatial relationship in an image thanks to involution kernels which are shared across channels; differently to convolution kernels which are limited to channels.

The train process is finally computed by combining the counterbalance of the vanilla distillation loss function ($\mathcal{L}_{Dstl}$) and the two just introduced attention distillation losses.

On a different application scenario, Lin et al. [38] (2023) focus on ViT models applied for few-shot learning (FSL) tasks on small datasets. Under these settings, ViT tends to overfit and suffers from severe performance degradation due to the high number of trainable parameters. Consequently, the authors focus on a KD learning strategy named supervised masked KD (SMKD), in which the student model only learns from a masked input sample, i.e., a reduced number of patches. Moreover, the authors introduce an intra-path loss function $\mathcal{L}_{patch}$ which compares the student (s) and teacher (t) embedding vectors by computing the cross-entropy loss of $s/t-$matching patches. The introduced strategy is finally completed by adding the vanilla KD learning procedure to the overall learning procedure.

3.4 Quantization

Based on the mathematical background introduced in Section 2.4, and motivated by the fact that ViTs are both memory and computation expensive during inference, this section reviews multiple researches focused on ViT quantization strategies specifically designed to reduce the costs of memory and computation. Similar to previous analysis, we review these efficient solutions for ViT architectures according to their release date. In particular, we will focus on general and hardware-specific quantization strategies designed to closely match ViT full precision data distribution with a lower bit-width.

Liu et al. [47] (2021) have been the first to explore post-training quantization for ViT architecture. The proposed compression strategy, based on mixed-precision weights, is formulated as an optimization problem without taking into account any training or fine-tuning process. Basically, the problem has the objective of finding the optimal low-bit quantization intervals for both weights and inputs in order to reduce both memory storage and computational costs. The authors focus on the FNN and MSA modules of ViT, aiming to assign the lowest possible bit-width to each attention module in order to maximize the prediction similarity between the full-precision model and the quantized one.

Similarly to Liu et al., Yuan et al. [79] (2022) propose an efficient framework for post-training quantization named PTQ4ViT. The main advantage of the proposed solution, with respect to the previous one, is the use of a twin uniform quantization strategy, which separately quantifies the negative and positive values in two ranges: R_1 and R_2, respectively. This choice is mainly due to the values achieved after softmax layers and GeLU activation functions of the attention block since their data distribution is very unbalanced and consequently very difficult to quantify. Therefore, based on Equation 5, and by defining the quantization intervals (scaling factors) Δ_{R_1} and Δ_{R_2} respectively for the two ranges R_1 and R_2, we can define the *k*-bit twin uniform quantization as reported in Eqution 23.

$$T(x, \Delta_{R_1}, \Delta_{R_2}) = \begin{cases} \Psi_{k-1}(x, \Delta_{R_1}) & \text{if } x \in R_1 \\ \Psi_{k-1}(x, \Delta_{R_2}) & \text{otherwise} \end{cases} \tag{23}$$

Moreover, in order to guide the optimal scaling factors for each layer for ViT, the authors propose to use the Hessian-guided metric to determine the quantization parameters.

However, Ding et al. [15] (2022), motivated by a notable accuracy drop at ultra-low bit-widths quantization, i.e., 4-bit, when Hessian-guided metric is employed to measure the quantization loss, propose a different approach named Accurate Post-training Quantization framework for Vision Transformer (APQ-ViT). Precisely, the authors focus on the design of a Matthew-effect[8] Preserving Quantization (MPQ) for the softmax function of the attention block. More in detail, APQ-ViT is composed of two phases: a quantization loss based on a unified Blockwise Bottom-elimination Calibration scheme to optimize the calibration metric and the MPQ specifically tailored for ViT models. Precisely, by defining the softmax function as $softmax(\cdot)$, the scaling factor (Δ_{MPQ}) can be defined as reported in Equation 24 and the MPQ function as reported in Equation 25.

$$\Delta_{MPQ} = \frac{max(softmax(\cdot))}{2^k - 1} \tag{24}$$

$$\Psi_k(\cdot,\cdot) = Clamp\bigg(Round\bigg(\frac{softmax(\cdot)}{\Delta_{MPQ}}\bigg), 0, 2^k - 1\bigg) \tag{25}$$

Moreover, Liu et al. [43] (2023) propose a plug-and-play quantizer-agnostic enhancement method for post-training ViT activation functions quantization. The proposed strategy, namely NoisyQuant, aims to improve previous quantization methods by adding a fixed NoisyBias (Nb) sampled from the Uniform distribution. Precisely, given the output distribution of the GeLU activation function (X), the NoisyQuant strategy can be obtained as $X + Nb$. This operation, computed before the quantization, flattens the data peaks, making the overall compression process more friendly. Moreover, the authors demonstrate that this sort of *soft-bounds* applied to the data distribution leads to a quantized output that closely follows the original data distribution with a lower bit rate.

Quantization methods are not only focused on compressing specific attention layers and respective activation functions, i.e., general ViT modules, in order to mimic original data distribution with lower precision data types; some works also focus on ViT compression strategies for specific hardware devices. This choice is mainly due to the real-world application of quantized models; this fact is particularly true in some scenarios like the binary compression proposed in Liu et al. [42] (EcoFormer). Precisely, when analyzing the limitations of the proposed method, the authors state that although EcoFormer is more efficient than previous solutions thanks to the binarization, in real-world applications, such as when inferring on GPU platforms, specific GPU kernel implementation, i.e., CUDA[9] operations would be required to take advantage of the proposed solution. This fact is due to the GPU device, which is unable to perform binary computations without allocating (in any case) a floating-point operation.

In this settings, Lit et al. [41] (2022) propose a framework, named Auto-ViT-Acc, which is specifically designed for quantizing ViT architectures to infer on FPGA[10] powered devices. The framework, which takes advantage of the quantization function introduced in [7], is applied only on FNN module of the attention block in order to increase the FPGA resource utilization and speed the inference process.

Moreover, differently from all the previous approaches, Yu et al. [74] (2023) propose GPUSQ-ViT, a GPU-friendly framework that incorporates multiple compression techniques. The proposed mixed strategy leverages the use of the knowledge distillation learning technique during a preliminary pruning phase and the subsequent aware-training quantization. Precisely, in a KD learning strategy, the full precision model, also used as a teacher, is first pruned with 16-bit floating point weights and then quantized to a fixed-point (precision) in order to obtain the final student model.

4 Efficient Vision Transformer Performances

In this section, we review and compare the estimation performances of previously described efficient ViT strategies. More in detail, following the four selected efficient categories (CA, P, KD, and Q), we compare all the models on the ImageNet [56] classification task. Precisely, all the reported values are extracted from the original papers and refer to the architectures trained end-to-end on the ImageNet1K dataset. The ImageNet1K dataset is composed of 1.3M of training and 50K validation images covering common 1K classes at a resolution of 224×224 pixels. Moreover, for the CA efficient strategy, we report into the supplementary material, Section 6.3, the obtained results also on two other datasets, i.e., the COCO [40] object detection and instance segmentation dataset, and the ADE20K [83] semantic segmentation datasets.

We evaluate the compared models with respect to the accuracy metric (Top-1) for the classification, average precision (AP), i.e., AP^{box} and AP^{mask} for the object detection and instance segmentation, and mean intersection over union (mIoU) for the semantic segmentation. Moreover, we also report into Tables 2 3 4[11] the data augmentation (DA) techniques used to train the compared models. More in detail, we identify as baseline DA strategy the one proposed in [61] (indicated as ✓), which includes Rand-Augment, Mixup, CutMix, and random erasing operations, excluding Repeated Augmentation and Stochastic Depth. In the tables, symbols ✓✓ and ✗ indicate cases with more or less DA techniques compared to the baseline. Furthermore, we report the number of trainable parameters (#Par.) and floating point operations (FLOPs) as efficiency metrics for AC, P, and KD strategies, as well as the weight/activation bit-widths (#Bit) and model size (Size) for Q methods, in order to quantify and evaluate the impact of the executed optimizations. Moreover, inspired by Li et al. [37], we introduce a metric that is able to measure how much a model is efficient with

8. In mixed-precision, the Matthew-effect [23] is the situation in which layers with higher bit-widths would be trained maturely earlier, while the others with lower bit-widths may never have the chance to express the desired function.

9. CUDA is a software platform that enables accelerated GPU computing on multiple operating systems.

10. An FPGA is a programmable logic device composed by an integrated circuit whose logical processing functionalities are programmable.

11. We report DA techniques for CA, P, and KD categories since Q methodologies are usually compared only on three pretrained models.

respect to a reference baseline; we define the Efficient Error Rate (EER) as reported in Equation 26.

$$EER = \frac{1}{||i||} \cdot \sum_{i} \left(\frac{M_i}{R_i} \right) \tag{26}$$

Where $i \in \{\#Par., FLOPs, ...\}$ is the set of efficiency metrics, M_i and R_i are respectively the $i-$values for the efficient model under analysis and for the reference one. Consequently, the more efficient the proposed model, with respect to the baseline model chosen for the specific task, the lower the EER value will be. Furthermore, in the supplementary material (Section 6.2), we present a more in-depth examination of the proposed EER metric in order to highlight its applicability and flexibility across diverse real-world tasks. Moreover, the EER metric is solely measured in the classification task since it is an application scenario in which all the identified efficient categories (CA, P, KD, and Q) are compared. More in detail, in the following tables, we report as percentage value the EER computed by choosing as baseline (R) the ViT-B/16. [16] architecture. This choice has been derived from the fact that the reference architecture is the shallower non-optimized model among the ViT milestones originally proposed by Dosovitskiy et al. and thus best approximates the efficient solutions under consideration in terms of #Pram., FLOPs, #Bit, and Size. Generally speaking, the ViT-B/16 model could be considered as the upper bound of the efficient/lightweight architectures distribution in analysis. Such architecture counts 86.6 million (M) of trainable parameters, 17.6 gigaflops (G) of operations at 32-bit precision, and a model size of 330 megabyte (MB). We underline that in order to have a fair and comprehensive understanding of previously analyzed methods with respect to the chosen evaluation metrics, models that have not been trained/tested according to the previously reported criteria, i.e., with different image resolutions or evaluation datasets, will not be present in the following tables; at the same time, all the values that are not given in the respective papers will be denoted with the $-$ symbol.

The rest of the section is organized as follows: Section 4.1, Section 4.2, Section 4.3, and Section 4.4 respectively reviews the estimation performances of CA, P, KD, and Q strategies. Finally, Section 4.5 compares all the strategies that best perform on the well-known classification scenario, i.e., over the 1K classes of the ImageNet benchmark dataset.

4.1 Results of Compact Architectures strategies

In this section, we compare the classification performances of compact architectures (CA) previously introduced in Section 3.1. Moreover, we report in the supplementary material Section 6.3 their estimation performances over other object detection, instance, and semantic segmentation tasks. We compare the classification performances of CA methods over the ImageNet1K dataset in Table 2[12]. Based on the reported accuracy values, we can notice that the MViTv2-L model has obtained the highest estimation performances, equal to 85.3%. In this survey, however, we are interested in efficient DL techniques, i.e., in solutions that aim to find the best trade-off between the model's accuracy and the introduced EER metric. Moreover, MViTv2-L is the model with the highest number of trainable parameters, FLOPs, and, consequently, the EER value. Therefore, the MViTv2-L model might be considered an outlier in comparison to all other reported methods when looking for the Paretian optimality[13] between accuracy and EER. Moreover, based on the same architecture, the Castling-MViTv2-B with an EER of $4, 2\times$ lower and similar estimation performances (85.0%) should be preferred.

Differently, in the application scenario in which hardware-constrained devices such as embedded single board PC are required, the Ortho-T model with only 3.9M of trainable parameters and also a restricted amount of FLOPs operations (0.7G) should be considered. Simi-

TABLE 2
Quantitative comparison of **CA** models on ImageNet1K classification dataset. The best results are in bold, and the best (trade-off) efficient model is highlighted in gray.

Model	#Par. [M]	FLOPs [G]	Top-1 [%]	EER [%]	DA
PVT-Tiny	13.2	1.9	75.1	13.0	
PVT-Small	24.5	3.8	79.8	24.9	✓
PVT-Medium	44.2	6.7	81.2	44.5	
PVT-Large	61.4	9.8	81.7	63.3	
Swin-T	29	4.5	81.3	29.5	
Swin-S	50	8.7	83.0	53.5	✓✓
Swin-B	88	15.4	83.5	94.5	
SOFT-Tiny	13	1.9	79.3	12.9	
SOFT-Small	24	3.3	82.2	23.2	
SOFT-Medium	45	7.2	82.9	46.4	✗
SOFT-Large	64	11.0	83.1	68.2	
SOFT-Huge	87	16.3	83.3	96.5	
PoolFormer-S12	12	1.9	77.2	12.3	
PoolFormer-S24	21	3.5	80.3	22.1	
PoolFormer-S36	31	5.1	81.4	32.4	✓
PoolFormer-M36	56	9.0	82.1	57.9	
PoolFormer-M48	73	11.8	82.5	75.7	
PVTv2-B2-LiSRA	22.6	3.9	82.1	24.0	✓
MViTv2-T	24	4.7	82.3	27.2	
MViTv2-S	35	7.0	83.6	40.1	✓
MViTv2-B	52	10.2	84.4	59.0	
MViTv2-L	218	42.1	**85.3**	245.5	
XCiT-T12/8 (SimA)	7	4.8	79.4	17.7	✓
XCiT-T12/16 (SimA)	26	4.8	82.1	28.6	
Ortho-T	**3.9**	0.7	74.0	**4.2**	
Ortho-S	24.0	4.5	83.4	26.6	✓
Ortho-B	50.0	8.6	84.0	53.3	
Ortho-L	88.0	15.4	84.2	94.5	
Flowformer	-	6.3	80.6	-	-
Castling-LeViT-128	10.5	**0.49**	79.6	7.4	
Castling-LeViT-192	12.7	0.82	81.3	9.6	
Castling-LeViT-256	22.0	1.4	82.6	16.7	
Castling-LeViT-384	45.8	2.9	83.7	34.7	
Castling-MViTv2-T	24.1	4.5	84.1	26.7	-
Castling-MViTv2-S	34.7	6.9	84.6	39.6	
Castling-DeiT-B	87.2	17.3	84.2	99.5	
Castling-MViTv2-B	51.9	9.8	85.0	57.8	
EfficientViT-B1	9.1	0.52	79.4	6.7	
EfficientViT-B3	49	4.0	83.5	39.6	✓
EfficientViT-L1	53	5.3	84.5	45.6	

12. Please refer to the supplementary material Section 6.4 for a graphical representation of the model's distributions regard to accuracy-EER values.

13. Paretian optimality/efficiency is an economic theory that describes a situation in which an improvement in the scenario of one variable cannot be achieved without negatively affecting another variable. Therefore, given our set of compared models, the Paretian efficiency is used to identify the model in which a higher accuracy cannot be reached without worsening the EER value, i.e., by establishing the best accuracy-EER trade-off.

larly, we can notice that also XCiT-T12/8 with SimA optimization, EfficientViT-B1, and Castling-LeViT-128 could be viable solutions for an hardware-constrained scenario. In particular, the two latter models, which only counts $0.52\text{G}/0.49\text{G}$ FLOPs, an EER of $6.7\%/7.4\%$, and estimation performances comparable to heavier models (Top-1 equal to $79.4\%/79.6\%$), could be valuable architectures in scenarios where real-time inference performances are needed.

Moreover, by comparing the two variants of PVT architectures, the contribution provided by the pooling operation, which is widely employed in many CA methods such as PoolFormer, PVTv2, and MViT, can be highlighted. More in detail, PVTv2, based on a linearized version of the SRA self-attention, originally introduced in PVT, is capable of achieving equivalent estimation performances of PVT-Medium with a comparable EER value of PVT-Sall.

Finally, for the CA efficient category, we identify the Paretian optimality between accuracy and the EER evaluation metric with the Castling-MViTv2-T model (even if no DA techniques are provided into the original study). Such architecture generates accurate estimations, i.e., a Top-1 accuracy of 84.1%, which is almost comparable (-1.2%) to the 85.3% of the heavier MViTv2-L but with an EER gain of $+9.2\times$. Moreover, when compared with models with similar EER values, such as Swin-T (29.5%), MViTv2-T (27.2%), and Ortho-S (26.6%), the identified architecture with an EER of 26.7%, obtains more accurate performances, with an average accuracy boost of $+1.8\%$.

4.2 Results of Pruning strategies

In this section, we compare the estimation performances of the pruning strategies, which are mainly introduced in Section 3.2 and formalized in Section 2.2. We remind that P strategies aim to minimize the number of active connections and neurons in a neural network by setting their weight to zero. Usually computed due to an overparametrization of DL architectures at training time, this process will not affect the number of model parameters (#Par.) since they will still be saved in the network graph even with a zeroed weight. In contrast, this strategy will be beneficial for the inference time by lowering the amount of (non-zeroed) multiplications required to generate the final prediction. Consequently, in the results obtained by the compared strategies, which are reported in Table 3, it is possible to notice that P methodologies are typically applied to well-known architectures such as Swin and DeiT, where the same number of parameters correlates to a lesser amount of FLOPs (please refer to Table 2 and Table 4 for their values). More in detail, taking the X-Pruner strategy as an example, it is possible to achieve an average reduction of 29.9% and 51.3% respectively on Swin and DeiT models with limited average accuracy (Top-1) reduction of -0.8% and -2.6%.

Based on the reported results, it can be noticed that more accurate strategies (Top-1 $\geq 83.5\%$) such as DP/DPS and DynamicViT take advantage of the LV-ViT [28] structure, which is, as reported in the original paper and even without pruning, able to achieve better and faster estimation than Swin and DeiT architectures. Furthermore, this fact can also be easily noticed when comparing PS and DPS over DeiT and LV-ViT at the same dimensionality class, i.e., S/S and B/M. However, in order to determine the Paretian optimality between accuracy and EER, we identify the DynamicViT pruning technique applied to the LV-ViT-S/07 architecture as the best trade-off among the analyzed efficient solutions. Moreover, as shown in Table 3, the DynamicViT-LV-S/07, which leverages a baseline (✓) DA strategy, is able to obtain high estimation performances (Top-1 $= 83.0\%$) while maintaining a low EER (28.9%) when compared to all the other reported methods. In contrast, from a purely efficient perspective, the X-Pruner strategy applied to the DeiT-Ti structure provides discrete performances with a very low EER, making it a suitable option in situations that require extremely restricted computational requirements.

TABLE 3
Quantitative comparison of **P** models on ImageNet1K classification dataset. The best results are in bold, and the best (trade-off) efficient model is highlighted in gray. FLOPs* values represent the amount of non-zeroed operations.

Model	#Par. [M]	FLOPs* [G]	Top-1 [%]	EER [%]	DA
VTP-DeiT-B ($\tau = 0.2$)	86.4	13.8	81.3	88.5	✓
VTP-DeiT-B ($\tau = 0.4$)	86.4	10.0	80.7	77.9	
PS-DeiT-S	22	2.6	79.4	20.1	✓✓
DPS-DeiT-S	22	2.4	79.5	19.5	
PS-DeiT-B	87	9.8	81.5	77.3	
DPS-DeiT-B	87	9.4	81.6	76.2	
PS-LV-ViT-S	26	4.7	82.4	32.8	
DPS-LV-ViT-S	26	4.5	82.9	32.3	
PS-LV-ViT-M	56	8.6	83.5	56.7	
DPS-LV-ViT-M	56	8.3	83.7	55.9	
NViT-T	6.9	1.3	76.2	7.6	✓
NViT-S	21	4.2	82.2	24.1	
NViT-H	30	6.2	82.9	34.9	
NViT-B	34	6.8	83.3	38.9	
X-Pruner-DeiT-Ti	**5**	**0.6**	71.1	**4.6**	✓✓
X-Pruner-DeiT-S	22	2.4	78.9	19.5	
X-Pruner-DeiT-B	87	8.5	81.2	74.1	
X-Pruner-Swin-T	29	3.2	80.7	25.8	
X-Pruner-Swin-S	50	6.0	82.0	45.9	
DynamicViT-LV-S/05	26.9	3.7	82.0	26.0	✓
DynamicViT-LV-S/07	26.9	4.6	83.0	28.9	
DynamicViT-LV-M/07	57.1	8.5	83.8	57.1	
DynamicViT-LV-M/08	57.1	9.6	**83.9**	60.2	

4.3 Results of Knowledge Distillation strategies

This section compares the estimation performances of student models trained under knowledge distillation learning techniques. The following analysis is mainly concerned with the understanding of how novel or existing architecture (student) can benefit from the knowledge of an external supervisor (teacher) in order to generate more accurate estimations without significantly increasing the #Par. and FLOPs. The findings obtained by compared models, which are mainly introduced in Section 3.3 and formalized in Section 2.3, are quantitatively reported in Table 4.

Based on the reported results, it can be noticed that from the preliminary studies of Touvron et al. in 2021, the introduced DeiT architecture has been gradually enhanced and optimized. More in detail, this evolution can be noticed in further depth with the Manifold Distillation and DearKD strategies proposed in 2022. These efficient techniques achieve an average accuracy improvement of $+1.5\%$ and $+1.7\%$, respectively, while keeping the number of trainable parameters and FLOPs constant. Moreover, in

TABLE 4
Quantitative comparison of student models trained with reviewed **KD** strategies on ImageNet1K classification dataset. The best results are in bold, and the best (trade-off) efficient model is highlighted in gray.

Model (student)	#Par. [M]	FLOPs [G]	Top-1 [%]	EER [%]	DA
DeiT-Ti	5	**1.3**	74.5	6.6	
DeiT-S	22	4.6	81.2	25.7	✓✓
DeiT-B	87	17.6	83.4	100.2	
Manifold-DeiT-Ti	5	**1.3**	76.5	6.6	
Manifold-DeiT-S	22	4.6	82.2	25.7	✓
Manifold-Swin-T	29	4.5	82.2	29.5	
TinyViT-5M	5.4	**1.3**	79.1	6.8	
TinyViT-11M	11.0	2.0	81.5	12.0	✓
TinyViT-21M	21.0	4.3	83.1	24.3	
DearKD-DeiT-Ti	5	**1.3**	77.0	6.6	
DearKD-DeiT-S	22	4.6	82.8	25.7	✓
DearKD-DeiT-B	87	17.6	**84.4**	100.2	
CivT-Ti	6	-	74.9	-	✓
CivT-S	22	-	82.0	-	
Mini-DeiT-Ti	**3**	**1.3**	72.8	**5.4**	
Mini-DeiT-S	11	4.7	80.7	19.7	
Mini-DeiT-B	44	17.6	83.2	75.4	✓
Mini-Swin-T	12	4.6	81.4	20.0	
Mini-Swin-S	26	8.9	83.6	40.3	
Mini-Swin-B	46	15.7	84.3	71.2	
DynamicViT-LV-S/05	26.9	3.7	82.0	26.0	
DynamicViT-LV-S/07	26.9	4.6	83.0	28.9	✓
DynamicViT-LV-M/07	57.1	8.5	83.8	57.1	
DynamicViT-LV-M/08	57.1	9.6	83.9	60.2	

the application scenario of extremely stringent computational constraints, the Mini-Deit-Ti model obtained with the MiniViT KD learning technique could be a valuable option with a very low EER and adequate classification performance. However, we identify as the most efficient model between the one compared in Table 4, the TinyViT-21M student , which is able to achieve accurate estimations (Top-1 $= 83.1\%$) with a limited EER equal to 24.3%. Precisely, when comparing TinyViT-21M to similar models such as DearKD-DeiT-S, DynamicViT-LV-S/05, and DynamicViT-LV-S/07, no other student designs achieve superior classification scores with a restricted EER value.

4.4 Results of Quantization strategies

In this section, we compare the estimation performances of general-purpose quantization strategies, which have been introduced in Section 3.4 and formalized in Section 2.4. As commonly reported in the reference papers, the different strategies are compared to the same architecture structure; precisely, we report the results obtained over ViT, DeiT, and Swin architectures, respectively, in Table 5, Table 6, and Table 7.

Differently from previous studies, these tables report the bit-width (#Bit) in which the models are compressed for both weights (W) and activation functions (A), as well as the size (Size) of the model after the quantization. This choice is due to the fundamental effect of the Q efficient strategy, which tries to preserve the estimation performances of originally trained models with a lower data precision, i.e., compressing their data from 32-bit to 8/6/4-bit. As a result, after the Q methodology, the number of trainable parameters (#Pram.) will remain constant, but the data precision of each neuron, and hence the total size of the model, will be reduced.

TABLE 5
Quantitative comparison of **Q** strategies applied to ViT models on ImageNet1K classification dataset. The best results are in bold, and the best (trade-off) efficient model is highlighted in gray. Values reported with * are estimated.

Model (ViT)	#Bit W	#Bit A	Size [MB]	Top-1 [%]	EER [%]
Liu_Q-ViT-B	6	6	64.8	75.2	19.2
Liu_Q-ViT-L	6	6	231.6	75.5	44.4
Liu_Q-ViT-B	8	8	86.5	76.9	25.6
Liu_Q-ViT-L	8	8	306.4	76.4	58.9
PTQ4ViT-ViT-S	6	6	16.5	78.6	11.9
PTQ4ViT-ViT-B	6	6	64.8	81.6	19.2
PTQ4ViT-ViT-S	8	8	22.2	81.0	15.8
PTQ4ViT-ViT-B	8	8	86.5	84.2	25.6
APQ-ViT-ViT-T	4	4	**2.9**	17.6	**6.3**
APQ-ViT-ViT-S	4	4	11.1	47.9	7.9
APQ-ViT-ViT-B	4	4	43.0	41.4	12.8
APQ-ViT-ViT-T	8	4	4.1*	38.6	9.9
APQ-ViT-ViT-S	8	4	16.5*	67.2	11.9
APQ-ViT-ViT-B	8	4	64.8*	72.5	19.2
APQ-ViT-ViT-T	4	8	4.1*	59.4	18.4
APQ-ViT-ViT-S	4	8	16.5*	72.3	11.9
APQ-ViT-ViT-B	4	8	64.8*	72.6	19.2
APQ-ViT-ViT-T	6	6	4.1	69.5	18.4
APQ-ViT-ViT-S	6	6	16.5	79.1	11.9
APQ-ViT-ViT-B	6	6	64.8	82.2	19.2
APQ-ViT-ViT-T	8	8	5.4	74.8	13.3
APQ-ViT-ViT-S	8	8	22.2	81.2	15.8
APQ-ViT-ViT-B	8	8	86.5	**84.3**	25.6
NoisyQuant-PTQ4ViT-ViT-S	6	6	16.5	78.6	11.9
NoisyQuant-PTQ4ViT-ViT-B	6	6	64.8	82.3	19.2
NoisyQuant-PTQ4ViT-ViT-S	8	8	22.2	81.1	15.8
NoisyQuant-PTQ4ViT-ViT-B	8	8	86.5	84.2	25.6

The first performed analysis regards the optimal bit-width in order to generate efficient and accurate models. Based on the data presented in the multiple tables, it can be noticed that an extreme quantization, i.e., 4-bit precision, results in poor accuracy. More in detail, the APQ-ViT quantization strategy applied to the ViT-T and DeiT-Ti architectures, which have the smaller model's sizes, produce a Top-1 accuracy equal to 17.6% and 47.9%, respectively. In contrast, the same architectures with higher (8/6) bit-width are able to achieve adequate estimation performances with a Top-1 up to 74.8% and 72.0% for the ViT-T and DeiT-Ti structures respectively, while still maintaining a restricted model size. Furthermore, by observing the heavier Swin models, as shown in Table 7, it can be noticed that even with an extreme quantization (4-bit), those architectures can guarantee good performance, with a Top-1 accuracy of 77.1% and 76.5% for APQ-ViT-Swin-S and APQ-ViT-Swin-B respectively. To summarize, severe quantization strategies applied on small models often result in significant performance degradation; consequently, it would be preferable to use 6/8-bit approaches in the case of shallower models while using 4-bit compression for deeper architectures. However, an extreme quantization strategy could benefit in applications where computational requirements are a bottleneck for ViT inference or on specific hardware, such as in the case of Google Coral TPUs[14].

Finally, in order to determine the Paretian efficiency between compared methodologies and architectures, we will first identify the best trade-off solution for each given table

14. https://coral.ai/products/

TABLE 6
Quantitative comparison of **Q** strategies applied to DeiT models on ImageNet1K classification dataset. The best results are in bold, and the best (trade-off) efficient model is highlighted in gray. Values reported with * are estimated.

Model (DeiT)	#Bit W	#Bit A	Size [MB]	Top-1 [%]	EER [%]
Liu$_Q$-DeiT-B	4	4	43.6	75.9	12.8
Liu$_Q$-DeiT-S	6	6	16.6	75.1	11.9
Liu$_Q$-DeiT-B	6	6	64.3	77.5	19.1
Liu$_Q$-DeiT-S	8	8	22.2	78.1	15.8
Liu$_Q$-DeiT-B	8	8	86.8	81.3	25.6
PTQ4ViT-DeiT-S	6	6	16.6	76.3	11.9
PTQ4ViT-DeiT-B	6	6	64.3	80.2	19.1
PTQ4ViT-DeiT-S	8	8	22.2	79.5	15.8
PTQ4ViT-DeiT-B	8	8	86.8	81.5	25.6
APQ-ViT-DeiT-Ti	4	4	**2.5**	47.9	**6.6**
APQ-ViT-DeiT-S	4	4	11.0	43.5	7.9
APQ-ViT-DeiT-B	4	4	43.6	67.5	12.8
APQ-ViT-DeiT-Ti	8	4	3.7*	56.3	9.9
APQ-ViT-DeiT-S	8	4	16.6*	41.3	11.9
APQ-ViT-DeiT-B	8	4	64.3*	71.7	19.1
APQ-ViT-DeiT-Ti	4	8	3.7*	66.7	9.9
APQ-ViT-DeiT-S	4	8	16.6*	77.1	11.9
APQ-ViT-DeiT-B	4	8	64.3*	79.5	19.1
APQ-ViT-DeiT-Ti	6	6	3.7	70.5	9.9
APQ-ViT-DeiT-S	6	6	16.6	77.8	11.9
APQ-ViT-DeiT-B	6	6	64.3	80.4	19.1
APQ-ViT-DeiT-Ti	8	8	5.0	72.0	10.1
APQ-ViT-DeiT-S	8	8	22.2	79.8	15.8
APQ-ViT-DeiT-B	8	8	86.8	**81.7**	25.6
NoisyQuant-PTQ4ViT-DeiT-S	6	6	16.6	77.4	11.9
NoisyQuant-PTQ4ViT-DeiT-B	6	6	64.3	80.7	19.1
NoisyQuant-PTQ4ViT-DeiT-S	8	8	22.2	79.5	15.8
NoisyQuant-PTQ4ViT-DeiT-B	8	8	86.8	81.4	25.6

TABLE 7
Quantitative comparison of **Q** strategies applied to Swin models on ImageNet1K classification dataset. The best results are in bold, and the best (trade-off) efficient model is highlighted in gray.

Model	#Bit W	#Bit A	Size [MB]	Top-1 [%]	EER [%]
PTQ4ViT-Swin-T	6	6	**21.7**	80.5	12.5
PTQ4ViT-Swin-S	6	6	37.5	82.4	15.0
PTQ4ViT-Swin-B	6	6	66.0	84.0	19.4
PTQ4ViT-Swin-T	8	8	29.0	81.2	16.9
PTQ4ViT-Swin-S	8	8	50.0	83.1	20.1
PTQ4ViT-Swin-B	8	8	88.0	85.1	25.9
APQ-ViT-Swin-S	4	4	25.0	77.1	**10.0**
APQ-ViT-Swin-B	4	4	44.0	76.5	12.9
APQ-ViT-Swin-S	8	4	37.5	80.6	15.0
APQ-ViT-Swin-B	8	4	66.0	82.0	19.4
APQ-ViT-Swin-S	4	8	37.5	80.6	15.0
APQ-ViT-Swin-B	4	8	66.0	81.9	19.4
APQ-ViT-Swin-S	6	6	37.5	82.7	15.0
APQ-ViT-Swin-B	6	6	66.0	84.2	19.4
APQ-ViT-Swin-S	8	8	50.0	83.2	20.1
APQ-ViT-Swin-B	8	8	88.0	**85.2**	25.9
NoisyQuant-PTQ4ViT-Swin-T	6	6	**21.7**	80.5	12.5
NoisyQuant-PTQ4ViT-Swin-S	6	6	37.5	82.8	15.0
NoisyQuant-PTQ4ViT-Swin-B	6	6	66.0	84.7	19.4
NoisyQuant-PTQ4ViT-Swin-T	8	8	29.0	81.2	16.9
NoisyQuant-PTQ4ViT-Swin-S	8	8	50.0	83.1	20.1
NoisyQuant-PTQ4ViT-Swin-B	8	8	88.0	**85.2**	25.9

and then establish the ideal optimal quantization strategy. Based on the reported data, we identify the NoisyQuant-PTQ4ViT quantization approach as the best trade-off option among compared ones. However, we emphasize that also APQ-ViT performs quite similarly to NoisyQuant-PTQ4ViT and may be considered a suitable alternative. Moreover, similarly to previous study, in order to ensure a Top-1 accuracy greater than 80.0%, we identify the ViT-B, DeiT-B, and Swin-B architectures as the best trade-off solution for each compared table adopting the 6-bit NoisyQuant-PTQ4Vi quantization strategy. However, we determined the NoisyQuant-PTQ4ViT-Swin-B methodology as the most efficient model among all analyzed solutions. Specifically, the latter architecture achieves a Top-1 accuracy equal to 84.7% with a model size of 66.0MB; in contrast, the ViT-B and DeiT-B variants obtain inferior classification performances of -2.3% and -4.0% with almost equal EER values and model sizes.

4.5 Overall classification performances

In this section, the best efficient strategies identified among the CA, P, KD, and Q categories are compared. Moreover, in order to provide a comprehensive and broad overview on limitations and estimation performances, we report in Table 8 an inter-category comparison. Precisely, we analyze each strategy under all the evaluation metrics used in the previous analysis, i.e., bit-width (#Bit), model size, number of parameters (#Par.), FLOPs, accuracy (Top-1) and the EER (EER$_{catg}$) value computed in the respective category. Furthermore, by employing in the Equation 26 the set of metrics $i \in \{\#Bit, Size, \#Par., FLOPs\}$, we generate a category-independent EER value (EER$_{all}$) that takes into account all the compared metrics.

Based on the reported results, it is possible to notice that DynamicViT-LV/07 and TinyViT-21M strategies are both obtained via a KD learning technique. More in detail, DynamicViT-LV/07 is a pruning strategy that leverages the use of KD to dynamically and progressively shrink the pruned-student model in order to minimize the performance reduction caused by the model's sparsification. Even so, TinyViT-21M (student) model achieves comparable classification performances ($\simeq 83\%$) with a smaller and lighter model, i.e., -5.9M of trainable parameters, -0.3G FLOPs and -23.6MB. Moreover, thanks to the memory-efficient KD strategy introduced by Wu et al., the TinyViT-21M strategy obtains the lower EER$_{all}$ value equal to 43.5% when computed across all the reported evaluation metrics.

In contrast, when TinyViT-21M is compared with the NoisyQuant-PTQ4ViT-Swin-B approach, it can be noticed that having a greater number of FLOPs computed with a lower precision (6-bit) can be beneficial for the model size and the estimation performances. In comparison to TinyViT-21M, NoisyQuant-PTQ4ViT-Swin-B achieves the maximum Top-1 accuracy of 84.7%, with a tiny model of 66.0MB with respect to TinyViT-21M (84.0MB). Precisely, the quantized strategy achieves a Top-1 boost of $+1.6\%$ with a model that is -21.4% smaller. However, as also underlined by Liu et al. (EcoFormer), the use of specific CUDA kernels or optimized interpreters may be necessary to maximize the effective performance of these quantization-based algorithms. Therefore, the NoisyQuant-PTQ4ViT-Swin-B may be an effective and efficient strategy for specific hardware setups.

Finally, the Paretian optimality between Top-1 accuracy and EER$_{all}$ is obtained by the CA strategy with the Castling-MViTv2-T model. As shown in Table 8, the latter architecture achieves optimal estimation performances (Top-1

TABLE 8
Quantitative comparison between best **CA**, **P**, **KD**, and **Q** identified strategies on ImageNet1K classification dataset. The best results are in bold, and the best (trade-off) efficient model is highlighted in gray. Values reported with * are estimated. EER_{catg} corresponds to the EER value computed in the respective category, while EER_{all} corresponds to the EER value computed over all the reported metrics.

Category	Model	#Bit W	#Bit A	Size [MB]	#Par. [M]	FLOPs [G]	Top-1 [%]	EER_{catg} [%]	EER_{all} [%]
CA	Castling-MViTv2-T	32	32	96.4	24.1	4.5	84.1	26.7	45.6
P + KD	DynamicViT-LV-S/07	32	32	107.6	26.9	4.6	83.0	28.9	47.4
KD	TinyViT-21M	32	32	84.0	**21.0**	**4.3**	83.1	24.3	**43.5**
Q	NoisyQuant-PTQ4ViT-Swin-B	**6**	**6**	**66.0**	88.0	15.4	**84.7**	**19.4**	56.5

$= 84.1\%$) while maintaining a suitable EER_{all}. Precisely, when compared with TinyViT-21M, the Castling-MViTv2-T model achieves an accuracy boost of $+1.1\%$ at the expense of an EER_{all} increment of 2.1%; differently, when compared with NoisyQuant-PTQ4ViT-Swin-B, the accuracy is reduced by -0.6%, and the EER_{all} is increased by $+10.9\%$. However, as can be seen from the reported results, the Castling-MViTv2-T model does not excel in any of the reported metrics (bold in Table 8); this is because the objective of this final analysis is not to identify the best strategy for making ViTs efficient but rather to identify the model that best balances efficiency and classification performances in general-purpose settings.

5 Discussions and Conclusion

In this paper, we survey the literature on efficient ViT strategies, one of the main bottlenecks of such architectures. The available techniques were classified into four categories: compact architecture (CA), pruning (P), knowledge distillation (KD), and quantization (Q). Moreover, we formalize their algorithms (Section 2), we review proposed techniques in order to highlight their strengths and weaknesses (Section 3), and finally, we compare the model/strategy performances over well-known benchmark datasets (Section 4) in order to determine the best trade-off strategy between estimation performances and efficiency. Furthermore, we introduce a novel evaluation metric named Efficient Error Rate (EER), which is used to give a general overview of the efficiency performances of reviewed models in comparison to a predefined and fixed non-efficient baseline solution.

In conclusion, based on what has been previously determined and analyzed so far, this last section will focus on open challenges and promising research directions that may improve the efficiency of general-purpose ViT models. The reviewed works are only one of the first steps toward developing effective and efficient solutions, leaving much room for future improvement. Therefore, we identified three potential major limitations, which are following discussed: application/tasks, evaluation metrics, and merging strategy.

Application/Task: As observed from the preceding analysis, many of the compared efficient approaches solely rely on testing their respective performances on a well-known classification task; in contrast, in CA optimizations, a variety of scenarios have been tested. Therefore, exploring these efficient architectures in different or more complex tasks is necessary. Furthermore, whether these architectures can generalize to other tasks or be robust to adversarial attacks are fundamental notions that still need to be investigated and explained.

Evaluation metrics: Based on the tables reported in Section 4 and on the reviewed works, the metrics used to determine the quality of proposed solutions, such as accuracy (Top-1/AP/mIOU), FLOPs, and number of trainable parameters, are evaluation metrics and architecture's values commonly used for general-purpose tasks. Consequently, since the development of efficient solutions aims to determine the best trade-off between accuracy and efficiency in order to apply ViTs in real-world scenarios, the development of specific benchmark tests based on resource-constrained devices could aid in comparing proposed solutions in both intra and inter-categories. More in detail, these metrics could focus on assessing the degradation effect caused by optimization strategies; this is because lowering the model's computational cost and associated resource demands typically comes at the expense of a performance decrease.

Merging strategies: The attention mechanism has played a crucial role in the development of accurate computer vision systems; however, the reduction/linearization of its computational cost has not been without performance consequences. According to the summary reported in Table 1, it is also visible that compared strategies mainly focus on determining the most efficient solution only focusing on a specific technique, i.e., CA, P, KD, or Q. In contrast, the ability to combine multiple techniques, as exemplified in DynamicViT by Rao et al., might successfully limit the accuracy loss of shallower architecture with respect to deeper one. As a result, since each efficient strategy leads to a reduction of ViT computational requirements, studying solutions that employ multiple efficient methodologies and determining the optimal way these solutions can be integrated may be an effective pathway for developing future efficient methodologies.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[2] Ondřej Bojar et al. "Findings of the 2016 conference on machine translation". In: *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*. 2016, pp. 131–198.

[3] Daniel Bolya et al. "Hydra Attention: Efficient Attention with Many Heads". In: *Computer Vision – ECCV 2022 Workshops*. Ed. by Leonid Karlinsky, Tomer Michaeli, and Ko Nishino. Cham: Springer Nature Switzerland, 2023, pp. 35–49. ISBN: 978-3-031-25082-8.

[4] Daniel Bolya et al. "Token Merging: Your ViT but Faster". In: *International Conference on Learning Representations*. 2023.

[5] Han Cai et al. "EfficientViT: Lightweight Multi-Scale Attention for High-Resolution Dense Prediction". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 17302–17313.

[6] Zhaowei Cai and Nuno Vasconcelos. "Cascade r-cnn: Delving into high quality object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6154–6162.

[7] Sung-En Chang et al. "Mix and match: A novel fpga-centric deep neural network quantization framework". In: *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE. 2021, pp. 208–220.

[8] Xianing Chen et al. "DearKD: data-efficient early knowledge distillation for vision transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12052–12062.

[9] Zhe Chen et al. "Vision Transformer Adapter for Dense Predictions". In: *arXiv preprint arXiv:2205.08534* (2022).

[10] Marcos V Conde et al. "Swin2SR: SwinV2 Transformer for Compressed Image Super-Resolution and Restoration". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2022.

[11] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. "On the Relationship between Self-Attention and Convolutional Layers". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=HJlnC1rKPB.

[12] Jifeng Dai et al. "Deformable convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 764–773.

[13] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[14] Mingyu Ding et al. "Davit: Dual attention vision transformers". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV*. Springer. 2022, pp. 74–92.

[15] Yifu Ding et al. "Towards Accurate Post-Training Quantization for Vision Transformer". In: *Proceedings of the 30th ACM International Conference on Multimedia*. 2022, pp. 5380–5388.

[16] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[17] Haoqi Fan et al. "Multiscale vision transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6824–6835.

[18] Daniel Y Fu et al. "Monarch Mixer: A simple sub-quadratic GEMM-based architecture". In: *arXiv preprint arXiv:2310.12109* (2023).

[19] Kai Han et al. "A survey on vision transformer". In: *IEEE transactions on pattern analysis and machine intelligence* 45.1 (2022), pp. 87–110.

[20] Zhiwei Hao et al. "Learning efficient vision transformers via fine-grained manifold distillation". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 9164–9175.

[21] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[22] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[23] Weijun Hong et al. "Dropnas: Grouped operation dropout for differentiable architecture search". In: *arXiv preprint arXiv:2201.11679* (2022).

[24] Holger H Hoos and Thomas Stützle. *Stochastic local search: Foundations and applications*. Elsevier, 2004.

[25] Huaibo Huang, Xiaoqiang Zhou, and Ran He. "Orthogonal Transformer: An Efficient Vision Transformer Backbone with Token Orthogonalization". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 14596–14607.

[26] Zhaoyang Huang et al. "Flowformer: A transformer architecture for optical flow". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*. Springer. 2022, pp. 668–685.

[27] Steven A Janowsky. "Pruning versus clipping in neural networks". In: *Physical Review A* 39.12 (1989), p. 6600.

[28] Zi-Hang Jiang et al. "All tokens matter: Token labeling for training better vision transformers". In: *Advances in neural information processing systems* 34 (2021), pp. 18590–18602.

[29] Ehud D Karnin. "A simple procedure for pruning back-propagation trained neural networks". In: *IEEE transactions on neural networks* 1.2 (1990), pp. 239–242.

[30] Angelos Katharopoulos et al. "Transformers are rnns: Fast autoregressive transformers with linear attention". In: *International conference on machine learning*. PMLR. 2020, pp. 5156–5165.

[31] Salman Khan et al. "Transformers in vision: A survey". In: *ACM computing surveys (CSUR)* 54.10s (2022), pp. 1–41.

[32] Alexander Kirillov et al. "Panoptic feature pyramid networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 6399–6408.

[33] Zhenglun Kong et al. "SPViT: Enabling Faster Vision Transformers via Latency-Aware Soft Token Pruning". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*. Springer. 2022, pp. 620–640.

[34] Soroush Abbasi Koohpayegani and Hamed Pirsiavash. "Sima: Simple softmax-free attention for vision transformers". In: *arXiv preprint arXiv:2206.08898* (2022).

[35] Duo Li et al. "Involution: Inverting the inherence of convolution for visual recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 12321–12330.

[36] Yanghao Li et al. "Mvitv2: Improved multiscale vision transformers for classification and detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 4804–4814.

[37] Yanyu Li et al. "Rethinking Vision Transformers for MobileNet Size and Speed". In: *arXiv preprint arXiv:2212.08059* (2022).

[38] Han Lin et al. "Supervised masked knowledge distillation for few-shot transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 19649–19659.

[39] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[40] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755.

[41] Zhengang Lit et al. "Auto-ViT-Acc: An FPGA-aware automatic acceleration framework for vision transformer with mixed-scheme quantization". In: *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE. 2022, pp. 109–116.

[42] Jing Liu et al. *EcoFormer: Energy-Saving Attention with Linear Complexity*. 2023. arXiv: 2209.09004 [cs.CV].

[43] Yijiang Liu et al. "NoisyQuant: Noisy Bias-Enhanced Post-Training Activation Quantization for Vision Transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 20321–20330.

[44] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[45] Ze Liu et al. "Swin transformer v2: Scaling up capacity and resolution". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 12009–12019.

[46] Ze Liu et al. "Swin transformer: Hierarchical vision transformer using shifted windows". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 10012–10022.

[47] Zhenhua Liu et al. "Post-training quantization for vision transformer". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 28092–28103.

[48] Zhuang Liu et al. "Learning efficient convolutional networks through network slimming". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2736–2744.

[49] Jiachen Lu et al. "Soft: Softmax-free transformer with linear complexity". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 21297–21309.

[50] Minh-Thang Luong and Christopher D Manning. "Stanford neural machine translation systems for spoken language domains". In: *Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign*. 2015, pp. 76–79.

[51] Lorenzo Papa, Paolo Russo, and Irene Amerini. "METER: a mobile vision transformer architecture for monocular depth estimation". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2023).

[52] Ali Rahimi and Benjamin Recht. "Random features for large-scale kernel machines". In: *Advances in neural information processing systems* 20 (2007).

[53] Yongming Rao et al. "Dynamic spatial sparsification for efficient vision transformers and convolutional neural networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[54] Yongming Rao et al. "Dynamicvit: Efficient vision transformers with dynamic token sparsification". In: *Advances in neural information processing systems* 34 (2021), pp. 13937–13949.

[55] Sucheng Ren et al. "Co-advise: Cross inductive bias distillation". In: *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*. 2022, pp. 16773–16782.

[56] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[57] Yehui Tang et al. "Patch slimming for efficient vision transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12165–12174.

[58] Yi Tay et al. "Efficient transformers: A survey". In: *ACM Computing Surveys* 55.6 (2022), pp. 1–28.

[59] Ilya O Tolstikhin et al. "Mlp-mixer: An all-mlp architecture for vision". In: *Advances in neural information processing systems* 34 (2021), pp. 24261–24272.

[60] Hugo Touvron et al. "Three Things Everyone Should Know About Vision Transformers". In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 497–515. ISBN: 978-3-031-20053-3.

[61] Hugo Touvron et al. "Training data-efficient image transformers & distillation through attention". In: *International conference on machine learning*. PMLR. 2021, pp. 10347–10357.

[62] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[63] Gary R Waissi. *Network flows: Theory, algorithms, and applications*. 1994.

[64] Wenhai Wang et al. "Pvt v2: Improved baselines with pyramid vision transformer". In: *Computational Visual Media* 8.3 (2022), pp. 415–424.

[65] Wenhai Wang et al. "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 568–578.

[66] Wenhui Wang et al. "Image as a Foreign Language: BEiT Pretraining for Vision and Vision-Language Tasks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 19175–19186.

[67] Xiaolong Wang et al. "Non-local neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7794–7803.

[68] Kan Wu et al. "Tinyvit: Fast pretraining distillation for small vision transformers". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXI*. Springer. 2022, pp. 68–85.

[69] Tete Xiao et al. "Unified perceptual parsing for scene understanding". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 418–434.

[70] Huanrui Yang et al. "Global Vision Transformer Pruning With Hessian-Aware Saliency". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 18547–18557.

[71] Hongxu Yin et al. "A-vit: Adaptive tokens for efficient vision transformer". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10809–10818.

[72] Hongxu Yin et al. "Dreaming to distill: Data-free knowledge transfer via deepinversion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8715–8724.

[73] Haoran You et al. "Castling-ViT: Compressing Self-Attention via Switching Towards Linear-Angular Attention at Vision Transformer Inference". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 14431–14442.

[74] Chong Yu et al. "Boost Vision Transformer with GPU-Friendly Sparsity and Quantization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 22658–22668.

[75] Fang Yu et al. "Width & Depth Pruning for Vision Transformers". In: *AAAI Conference on Artificial Intelligence*. 2022.

[76] Lu Yu and Wei Xiang. "X-Pruner: eXplainable Pruning for Vision Transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 24355–24363.

[77] Weihao Yu et al. "Metaformer is actually what you need for vision". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10819–10829.

[78] Weihao Yuan et al. "Neural window fully-connected crfs for monocular depth estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 3916–3925.

[79] Zhihang Yuan et al. "Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XII*. Springer. 2022, pp. 191–207.

[80] Xiaohua Zhai et al. "Scaling vision transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12104–12113.

[81] Jinnian Zhang et al. "Minivit: Compressing vision transformers with weight multiplexing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12145–12154.

[82] Li Zhang et al. "Dynamic graph message passing networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 3726–3735.

[83] Bolei Zhou et al. "Scene parsing through ade20k dataset". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 633–641.

[84] Mingjian Zhu, Yehui Tang, and Kai Han. "Vision transformer pruning". In: *arXiv preprint arXiv:2104.08500* (2021).

[85] Bohan Zhuang et al. "A Survey on Efficient Training of Transformers". In: Aug. 2023, pp. 6823–6831. DOI: 10.24963/ijcai.2023/764.

LORENZO PAPA (Student Member, IEEE) is a Ph.D. student in Computer Science Engineering. He collaborates with AlcorLab at the Department of Computer, Control, and Management Engineering, Sapienza University of Rome, Italy. He is a Visiting Researcher at the School of Electrical and Information Engineering, Faculty of Engineering, The University of Sydney, Australia. He received the B.S. degree in Computer and Automation Engineering and the M.S. degree in Artificial Intelligence and Robotics from Sapienza University of Rome, Italy, in 2019 and 2021, respectively. His main research interests are Deep Learning, Computer Vision, and Cyber Security.

PAOLO RUSSO is an Assistant Researcher at AlcorLab in DIAG department, University of Rome Sapienza, Italy. He received the B.S. degree in Telecommunication Engineering from Università degli studi di Cassino, Italy, in 2008, and the M.S. degree in Artificial Intelligence and Robotics from University of Rome La Sapienza, Italy, in 2016. He received Ph.D. degree in Computer Science from University of Rome La Sapienza in 2020. From 2018 to 2019, he has been a researcher at Italian Institute of Technology (IIT) in Tourin, Italy. His main research interests are Deep Learning, Computer Vision, Generative Adversarial Networks, and Reinforcement Learning.

IRENE AMERINI (M'17) received Ph.D. degree in computer engineering, multimedia, and telecommunication from the University of Florence, Italy, in 2010. She is currently Associate Professor with the Department of Computer, Control, and Management Engineering A. Ruberti, Sapienza University of Rome, Italy. Her main research activities include digital image processing, computer vision multimedia forensics. She is a member of the IEEE Information Forensics and Security Technical Committee and the EURASIP TAC Biometrics, Data Forensics, and Security, and the IAPR TC6 - Computational Forensics Committee.

LUPING ZHOU (Senior Member, IEEE) received the Ph.D. degree from Australian National University. She is currently an Associate Professor with the School of Electrical and Information Engineering, The University of Sydney, Australia. Her research interests include medical image analysis, machine learning, and computer vision. She was a recipient of Australian Research Council Discovery Early Career Researcher (DECRA) Award in 2015.