Neural Networks 194 (2026) 108082

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Full Length Article

Adaptive behavior with stable synapses

Cristiano Capone [a,1], Luca Falorsi [a,b,1,*]

[a] Natl. Center for Radiation Protection and Computational Physics, Istituto Superiore di Sanità, Viale Regina Elena 299, Rome, RM, 00161, Italy
[b] PhD program in Mathematics, Sapienza University of Rome, Piazzale Aldo Moro 5, Rome, RM, 00185, Italy

ARTICLE INFO

Keywords:
In context learning
Dynamical learning
Gain modulation
Reservoir computing
Hierarchical learning
Brain-inspired computation

ABSTRACT

Behavioral changes in animals and humans, triggered by errors or verbal instructions, can occur extremely rapidly. While learning theories typically attribute improvements in performance to synaptic plasticity, recent findings suggest that such fast adaptations may instead result from dynamic reconfiguration of the networks involved without changes to synaptic weights. Recently, similar capabilities have been observed in transformers, foundational architecture in machine learning widely used in applications such as natural language and image processing. Transformers are capable of in-context learning, the ability to adapt and acquire new information dynamically within the context of the task or environment they are currently engaged in, without changing their parameters. We argue that this property may stem from gain modulation–a feature widely observed in biological networks, such as pyramidal neurons through input segregation and dendritic amplification. We propose a constructive approach to induce in-context learning in an architecture composed of recurrent networks with gain modulation, demonstrating abilities inaccessible to standard networks. In particular, we show that, such architecture can dynamically implement standard gradient-based by encoding weight changes in the activity of another network. We argue that, while these algorithms are traditionally associated with synaptic plasticity, their reliance on non-local terms suggests that they may be more naturally realized in the brain at the level of neural circuits. We demonstrate that we can extend our approach to temporal tasks and reinforcement learning. We further validate our approach in a MuJoCo ant navigation task, showcasing a neuromorphic control paradigm via real-time network reconfiguration.

1. Introduction

Adaptive behavior in biological agents. The study of adaptive behavior in biological agents, including humans and animals, has been a long-standing subject of research (Lee & Lee, 2013). Observations of rapid behavioral shifts in response to environmental cues have raised questions about whether traditional mechanisms, such as synaptic plasticity, can fully account for such flexibility.

A broad range of neural and behavioral data indicates the existence of distinct systems for behavioral choice (Daw et al., 2005; McDougle et al., 2015). The conventional view posits that subcortical areas support habitual, stimulus-based responses, enabling reflex-based and rapid reactions. In contrast, higher cortical areas, such as the prefrontal cortex, are associated with reflective, goal-directed behavior, which relies on cognitive action planning and deliberate evaluation. Critically, goal-directed behavior allows for the flexible construction of value at each decision point through online planning procedures. This mechanism makes agents immediately sensitive to changes in outcome values, enabling them to dynamically adapt their actions to novel or changing circumstances (O'Doherty et al., 2017). Studies in network neuroscience have identified dynamic network reconfiguration, characterized by the flexible recruitment and integration of neural circuits, as a fundamental mechanism supporting this adaptability across cognitive (Braun et al., 2015) and motor domains (Standage et al., 2023).

Computational models within the predictive coding framework have recently provided insights into the neural mechanisms underlying dynamic adaptation Millidge et al. (2024). These models suggest that adaptation can be understood as a form of Bayesian inference, where the brain continuously infers latent contextual states to guide behavior (Heald et al., 2023b). A prominent example is the COIN model (Heald et al., 2021, 2023a), which posits that adaptation arises through two complementary processes: proper learning, involving the creation and updating of memories, and apparent learning, which dynamically adjusts the expression of pre-existing memories based on inferred context. While these frameworks capture key phenomena of adaptive behavior, such as context-dependent single-trial learning and spontaneous

* Corresponding author.
E-mail addresses: cristiano.capone@iss.it (C. Capone), luca.falorsi@gmail.com (L. Falorsi).
[1] These authors contributed equally.

https://doi.org/10.1016/j.neunet.2025.108082
Received 17 April 2025; Received in revised form 7 August 2025; Accepted 2 September 2025
Available online 6 September 2025
0893-6080/© 2025 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

recovery, a significant gap remains between these theoretical models and the neural substrates that implement these computations in the brain.

In-context learning as an emerging property in AI. In machine learning, the capacity of a model to adapt its behavior, without weight updates, is referred to as in-context learning (ICL). Initially, ICL was observed in architectures tailored for few-shot learning (Vinyals et al., 2016) or even zero-shot learning (Romera-Paredes & Torr, 2015). However, the game changed when it was observed that ICL emerges naturally in large-scale transformers (Brown et al., 2020; Singh et al., 2024). These architectures leverage self-attention, a mechanism that dynamically changes the way past inputs are attended, in order to shape the current response. Their exceptional capability to adapt to contextual information allowed transformer-based architectures to achieve state-of-the-art performances in many domains, such as natural language processing and image analysis (Achiam et al., 2023; Ramesh et al., 2021; Vaswani et al., 2017). While there are several intriguing explanations (Olsson et al., 2022; Ramsauer et al., 2020) for the emergence of ICL in transformers, a more formal understanding was pioneered by Von Oswald and collaborators (Von Oswald et al., 2023), who demonstrated that a linear self-attention layer can implement gradient descent steps within its forward pass. This intuition was empirically validated by analyzing trained transformers and was formally proven (Zhang et al., 2024).

Despite their success, transformer architectures face notable shortcomings, particularly in their memory requirements, which scale quadratically with sequence length. These requirements make those architectures largely biologically implausible. Moreover, memory demands constrain their scalability and restrict their applicability to tasks involving long sequences, such as full-length document analysis or video understanding. To overcome these challenges, several efficient deep linear RNN architectures have been proposed (Gu et al., 2021; Orvieto et al., 2023), demonstrating substantial performance gains over transformers, especially in long-sequence tasks.

However, these deep learning architectures rely on several mechanisms that do not have an obvious correspondence in biological neural systems (such as the attention mechanism itself), leaving an unresolved gap in relating them to the in-context learning abilities exhibited by biological networks. This work bridges this gap by introducing a constructive method to induce in-context learning in biologically plausible neural networks, demonstrating its applicability in a wide range of scenarios.

Gain-modulation for dynamic behavioral adaptation. There is strong experimental and theoretical support for gain modulation as a mechanism for rapid adaptation, independent of synaptic rewiring. Dendritic amplification, for instance, shows that top-down inputs to apical dendrites of layer 5 pyramidal neurons can instantly increase neuronal gain and alter firing patterns, enabling rapid association of inputs without synaptic changes (Guarino et al., 2025; Larkum, 2013a). Computational models further demonstrate that gain changes alone can rapidly shift network dynamics and behavior without requiring synaptic modifications (Köksal-Ersöz et al., 2024), a possibility supported by experimental findings involving optogenetic disinhibition and dendritic amplification (Cardin et al., 2008; Ferguson & Cardin, 2020). In the realm of sensorimotor adaptation, human motor learning experiments reveal that feedback gains are rapidly tuned to new environments or tasks within a single session, allowing for flexible control without relying on synaptic plasticity (Heald et al., 2021). More recently, in Wainstein et al. (2025) authors provided direct evidence that phasic neuromodulatory bursts dynamically modulate neural gain to mediate rapid perceptual switches; through pupillometry, fMRI, and recurrent neural network modeling, they show that heightened gain transiently destabilizes neural dynamics during periods of uncertainty, thereby accelerating perceptual updating. In Stroud et al. (2018), the authors also discuss how the cerebellum modulates motor cortex activity in response to error signals. Taken together, these findings offer strong biological grounding for gain modulation as a fast and flexible learning mechanism, supporting the framework presented in this manuscript.

Biological support for in-context learning. Building on this premise, our investigation explores how in-context learning, as demonstrated in deep learning architectures, might also be realized in biological recurrent neural networks thanks to gain modulation. Are there unique features in transformers that are also present in biological networks?

We propose that in-context learning can be facilitated by input segregation and dendritic amplification, two features widely observed in biological networks. We further argue that these mechanisms provide a biologically plausible foundation for implementing a process analogous to the attention mechanism in transformers. Recent findings on dendritic computational properties (Poirazi & Papoutsi, 2020) and on the complexity of pyramidal neurons dynamics (Larkum, 2013a) motivated the study of multi-compartment neuron models in the development of new biologically plausible learning rules (Guerguiev et al., 2017; Payeur et al., 2021; Sacramento et al., 2018; Urbanczik & Walter, 2014). It has been proposed that segregation of dendritic input (Guerguiev et al., 2017) (i. e., neurons receive sensory information and higher-order feedback in segregated compartments) and generation of high-frequency bursts of spikes (Payeur et al., 2021) would support backpropagation in biological neurons.

Recently, it has been suggested (Capone et al., 2023) that this neuronal architecture naturally allows for orchestrating “hierarchical imitation learning”, enabling the decomposition of challenging long-horizon decision-making tasks into simpler subtasks. They show a possible implementation of this in a two-level network, where the high-network produces the contextual signal for the low-network. Here, we propose an architecture composed of gain-modulated recurrent networks that demonstrate remarkable in-context learning capabilities, which we refer to as ‘dynamical adaptation’. Specifically, we illustrate that our biologically plausible architecture can dynamically adapt its behavior in response to feedback from the environment without altering its synaptic weights. We present results for supervised learning of temporal trajectories and reinforcement learning, involving non trivial input-output temporal relations. This novel architecture aims to bridge the gap between biological-inspired in-context learning and the capabilities of artificial neural networks, offering a promising avenue for advancing our understanding of adaptive behavior in both natural and artificial intelligence domains.

2. Main

2.1. Framework formalization

Consider a general learning scenario, where an agent is engaged in an environment α from a family of tasks $\mathcal{A}$. The agent receives temporal input $\mathbf{x}(t)$ and it has to compute a response $y(t)$, receiving feedback $f(t)$ from the environment in return. This can be an error, reward, or the target behavior.

We further assume that a low-level network extracts features $\mathbf{z}(t)$ relevant to the task family. The agent then adapts its response to the specific environment by learning readout parameters $\mathbf{w}$, which determine how to recombine these features to produce the correct response, $y(t) = Y(\mathbf{z}(t), \mathbf{w})$. Here Y is a generic function that depends on the parameters $\mathbf{w}$. These parameters are then adjusted at each time step in function of an internally computed error signal e. The overall learning process can then be generically rewritten as the following dynamical system:

$$\begin{cases} \tau_e \dot{\mathbf{e}} & = E(\mathbf{e}, \mathbf{y}, \mathbf{f}), \\ \tau_y \mathbf{y} & = Y(\mathbf{y}, \mathbf{w}, \mathbf{z}), \\ \tau_w \dot{\mathbf{w}} & = W(\mathbf{w}, \mathbf{z}, \mathbf{e}). \end{cases} \tag{1}$$

The specific instantiation of the functions W and E depends on the learning algorithm employed and can encompass both supervised and reinforcement learning settings. Refer to the Methods section for their precise definitions. In the classic view, the parameters $\mathbf{w}$ are interpreted as synaptic weights of a neuronal network, and the functions E and W describe synaptic plasticity mechanisms. However, this imposes stringent locality requirements, as all information must be present at the synapse.

Rather than interpreting $\mathbf{w}$ as synaptic weights, we view them as the dynamic activity of an auxiliary network that modulates the response Y in real time based on feedback and past interactions. We refer to these dynamically adapting state variables as "virtual" weights. We refer to $\mathbf{w}$ as "virtual weights" because: - Weights: they replace the physical weights, and they have the same role as the physical weights of parametrizing the map from the features $\mathbf{z}(t)$ to the output(s). - Virtual: they are technically not weights, since the are encoded in the activity of the readout neurons of the network W, and their activity is provided as an input current to the Y network. Using this strategy we replace the computation done by the synapses (including learning) with operations performed by the neural activity. Dynamic adaptation offers key advantages over synaptic plasticity:

- **Fast timescales**: It naturally operates on faster timescales, making it well-suited for rapid behavioral changes
- **Non-locality**: Dynamic adaptation is not restricted by the stringent locality requirements of synaptic plasticity, allowing for richer forms of interaction and computation at the network level.

In the Supplementary Materials, we show that a non-local weight update rule, implemented through dynamic adaptation, enables rule switching with a single error feedback in a context-dependent classification task, whereas local synaptic plasticity drops to chance level after one error–independently of the timescales of the learning process (Fig. A.2). In the dynamic adaptation setting, the functions E and W in Eq. (1) arise from the dynamics of the auxiliary network of virtual weights. We show that such networks can dynamically implement standard gradient-based algorithms widely used in machine learning. While these algorithms are traditionally associated with synaptic plasticity, their reliance on non-local terms suggests that, in the brain, they can be more naturally realized at the circuit level. We demonstrate that such circuits capable of dynamic learning can simply arise by tuning the static readout weights of random RNNs with multiplicative interactions, requiring only a limited number of training trajectories from the original learning algorithm applied to a subset of the task family. The construction of our networks follows two distinct phases: a developmental or evolutionary phase, and a subsequent phase where the network leverages this structure for dynamical learning.

- **Phase 1: development of the network structure.** In this phase, using an Algorithmic Distillation protocol (Laskin et al., 2022), we collect learning trajectories of the form $\{(\mathbf{z}_\alpha(t), \mathbf{w}_\alpha(t), \mathbf{e}_\alpha(t), \mathbf{f}_\alpha(t))_{t=0}^{T} \| \alpha \in \mathcal{A}_{ID}\}$ by running the learning dynamics in Eq. (1) on a subset of tasks $\mathcal{A}_{ID}$ (In Distribution). We then train the static readout weights Θ of two random RNNs Y_{Θ_y} and W_{Θ_w} to replicate, respectively, the functions Y and W on the training data. This is done by minimising the losses:

$$L_Y(\Theta_y) = \sum_{\alpha,t} \left| Y_{\theta_y}(\{\mathbf{z}_\alpha(t), \mathbf{w}_\alpha(t)\}) - Y(\mathbf{z}_\alpha(t), \mathbf{w}_\alpha(t)) \right|^2$$

$$L_W(\Theta_w) = \sum_{\alpha,t} \left| W_{\Theta_w}(\{\mathbf{z}_\alpha(t), e_\alpha(t)\}) - W(\mathbf{z}_\alpha(t), e_\alpha(t)) \right|^2$$

- **Phase 2: the whole architecture is tested (dynamical learning).** The architecture is tested on a new, previously unseen task $\alpha \in \mathcal{A}_{OOD} = \mathcal{A} \setminus \mathcal{A}_{ID}$ (Out-of-Distribution). During this phase, all synaptic weights remain fixed. Instead of relying on synaptic plasticity, the virtual weights evolve dynamically according to: $\mathbf{w}(t + dt) = \mathbf{w}(t) + W_{\Theta_w}(t)$. The updated output of the network is given by $y(t) = Y_{\Theta_y}(\{\mathbf{w}(t), \mathbf{z}(t)\})$. Importantly, we remark that $\mathbf{w}(t)$ no longer represents physical synaptic weights but rather contextual signals encoded in the activity of neurons, acting as input currents to the Y_{Θ_y} network.

2.2. The role of gain-modulation in adaptive behavior

Recent research has highlighted a growing interest in understanding modulatory mechanisms and their role in neural computation. It has been observed that gain modulation allows neurons to adjust their response to inputs based on context (Larkum, 2013a). This dynamic regulation of neuronal sensitivity plays a critical role in adaptive behavior and cognitive flexibility. Moreover, it has been implicated in various functions, including feedback control (Feulner et al., 2022; Meulemans et al., 2022) and attention, where it allows for the selective amplification of relevant information. A recent study (Jiang & Rao, 2024) identified gain modulation as a key neural mechanism underlying hierarchical spatiotemporal prediction and sequence learning, demonstrating that such models can replicate various cortical processing features, including temporal abstraction and space-time receptive fields in the primary visual cortex. Furthermore, gain modulation is closely tied to meta-learning and multitask learning (Perez et al., 2018; Wybo et al., 2023), enabling neural networks to efficiently generalize across diverse tasks by leveraging contextual cues.

Within biological recurrent networks, dendritic segregation provides a structural basis for gain modulation. By compartmentalizing synaptic inputs, neurons can effectively implement non-linear interactions between different input streams. This is reminiscent of In-Context Learning (ICL) in machine learning, where multiplicative interactions are crucial for enabling models to leverage previously learned information in new contexts. Although we focus on gain modulation mediated by apical dendrites, other mechanisms are also known. For instance, neuro-glia interactions at the synapse have been proposed as a substrate for working memory, with astrocytes playing a crucial role in modulating neural activity and maintaining memory through glia–synapse interactions (De Pittà & Brunel, 2022). This gain modulation mechanism could complement other adaptive processes, such as subtle, correlated changes in connectivity (Feulner et al., 2022).

Our work explores the role of gain modulation in recurrent networks, employing a simplified model where apical currents act multiplicatively on basal currents within a transfer function. A key parameter, γ, controls the strength of these non-linear interactions (see Methods for further details). By modulating the gain of neuronal responses, these networks exhibit enhanced adaptability and in-context learning capabilities.

3. Results

3.1. Dynamical adaptation for temporal trajectories

We consider the task of autonomously predicting a temporal trajectory $\{f(t) = y^{targ}(t)\}_t$. Following the reservoir computing paradigm (see Methods section), we employ a random RNN (Fig. 1A, pink), whose dynamics follows Eq. (5), to extract temporal features $\mathbf{z}(t)$. During training, the reservoir receives the target dynamics $y^{targ}(t)$ as input ($x(t) = y^{targ}(t)$ in Eq. (5)) and is tasked with predicting the subsequent step of the trajectory via a linear readout of its activity (open loop, see Fig. 1B). This setup can be reformulated as a dynamical supervised learning problem, as described in the methods section. The readout weights can then be dynamically adjusted, minimizing the error between the target and the current prediction $y(t)$. This results in the following dynamics:

$$\begin{cases} e & = (y^{targ} - y) \\ y & = Y_{\Theta^y}(\{\mathbf{z}, \mathbf{w}\}) \simeq \mathbf{z} \cdot \mathbf{w} \\ \tau_w \dot{\mathbf{w}} & = W_{\Theta^w}(\{\mathbf{z}, e\}) \simeq \mathbf{z}\, e \end{cases} \tag{2}$$

Here, the network $W_{\Theta^w}(\{\mathbf{z}, e\})$ is dedicated to estimating the gradient of virtual weights, while $Y_{\Theta^y}(\{\mathbf{z}, \mathbf{w}\})$ is tasked with estimating the

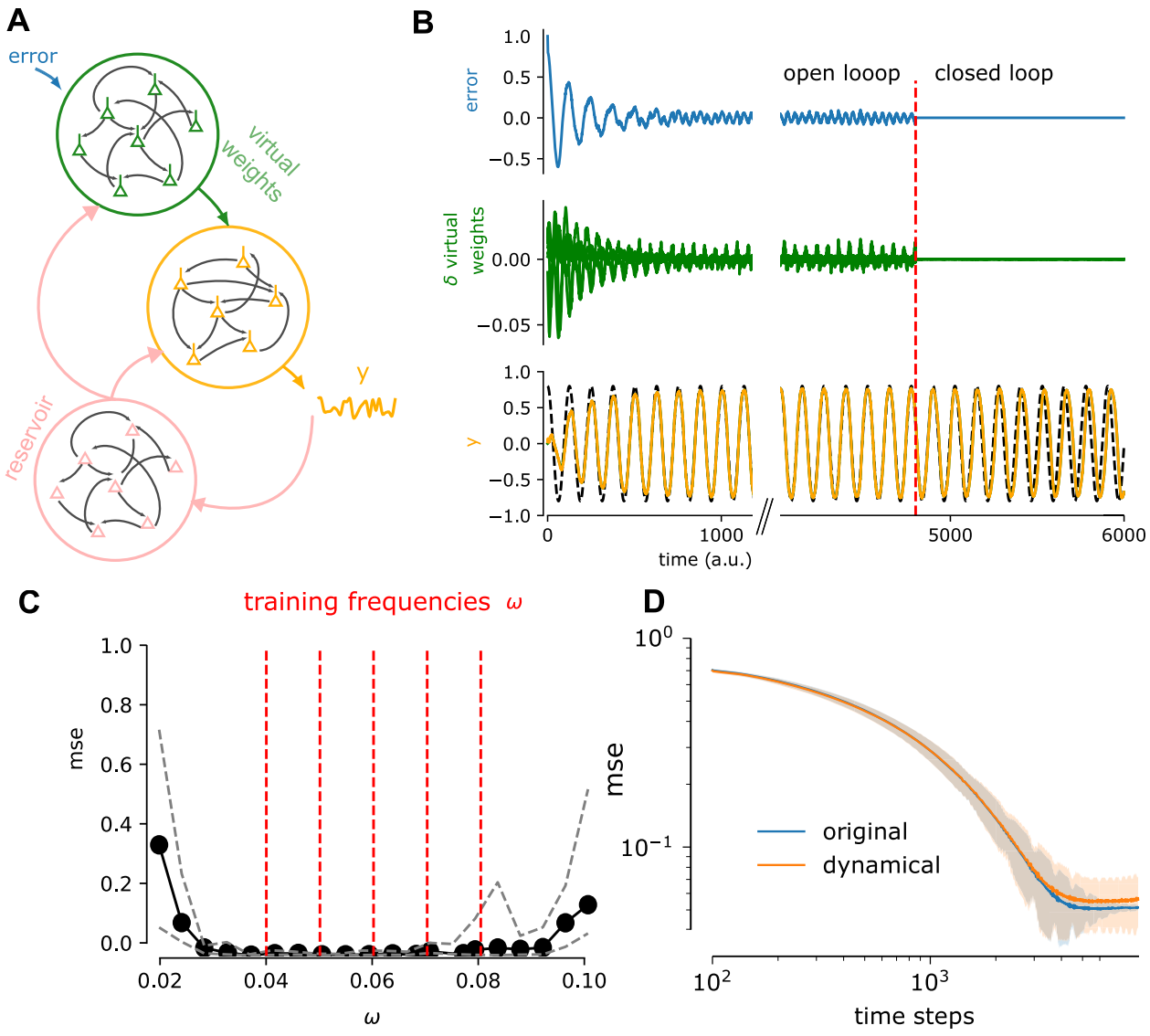

Fig. 1. Dynamical adaptation of temporal trajectories: A. Overview of the network architecture employed. A recurrent network composed of $N = 20$ units (described by the vector $\mathbf{z}(t)$, depicted in pink, which follows Eq. (5)) is utilized for learning a periodic trajectory, following the prescription of reservoir computing. One recurrent network is tasked with estimating the gradient of virtual weights, while another is dedicated to estimating the behavioral reconfiguration (illustrated in green and orange, respectively) resulting from these updated virtual weights. **B.** Illustration of our architecture dynamically adjusting (without synaptic alterations) to adhere to the desired dynamics. Errors (represented by the blue trajectory) are fed back to the initial network to assess necessary updates to the virtual weights δw, a N-dimensional vector (shown in the green trajectory). Subsequently, these updates are transmitted to the second network, modifying the decoding of reservoir dynamics (indicated by the orange lines). Initially, the reservoir receives the target trajectory as input (in open loop, before the red vertical line), which is later replaced by the estimated trajectory itself (in closed loop, after the red vertical line). **C.** Our networks were pre-trained on five target frequencies (marked by red vertical lines) and tested across a range of frequencies, evaluating the mean squared error (MSE) between the target and estimated trajectories in closed-loop scenarios (solid: median, dashed: 20th-80th percentile (statistics evaluated over 10 realizations). **D.** Comparison of the learning curve, mse in function of the weights update, for the original learning algorithm, and the one evaluated by the two networks, in blue and orange respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

predicted $y(t)$ as a function of the new virtual weights (Fig. 1A green and orange respectively).

These gain-modulated architectures are pre-trained to replicate, respectively, gradient descent updates and scalar products obtained on a set of N_{train} training sequences $\{\{y_\alpha^{targ}(t)\}_t : \alpha \in [N_{train}]\}$. More specifically, the target sequences are 5 sinusoidal functions with different frequencies (see vertical red lines in Fig. 1C.)

In the closed-loop phase, the features $\mathbf{z}$ are obtained by directly feeding the network estimation $y(t)$ as the input to the RNN ($x = y(t)$ in Eq. (5)). This results in an autonomous dynamical system that reproduces the target trajectory. (see Fig. 1B). In Fig. 1B we report an example of successful dynamical adaptation of our model. Errors, represented by the blue trajectory, are fed back to the initial network to assess necessary updates to virtual weights, shown in the green trajectory. These updates are then transmitted to the second network, modifying the decoding of reservoir dynamics indicated by the orange lines. Initially, the reservoir receives the target trajectory as input in an open-loop fashion before the red vertical line, which is later replaced by the estimated trajectory itself in a closed-loop configuration after the red vertical line. The networks were pre-trained on five target frequencies marked by red vertical lines in Fig. 1C and then tested across a range of frequencies. Evaluation was performed by assessing the mean squared

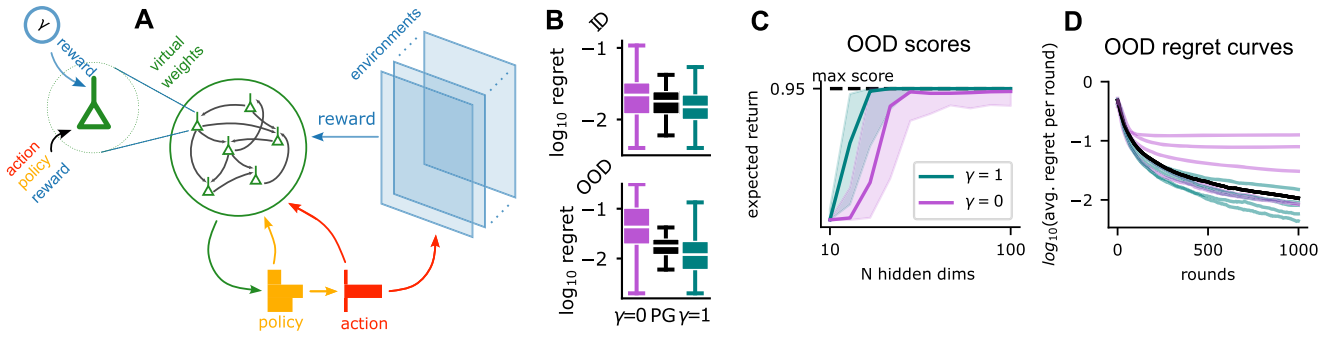

Fig. 2. Dynamical reinforcement learning: multi armed bandits. **A.** Schematic of network architecture and task. Virtual weighs parameterize the agent's policy (orange). At each round, an action (red) is sampled from the policy and played. A reward is then sampled from the current environment. A GM-network (green) then predicts the virtual parameter update. **B.** Regret comparison between the distilled and the original policy gradient algorithm. We report log regret per round distribution achieved at the 100-th round for 100 independently trained models in ID and OOD settings. We compare a gain-modulated network ($\gamma = 1$), and a network without gain modulation ($\gamma = 0$) with the policy gradient training source (PG). **C.** OOD model performance varying number of hidden dimensions. Solid lines indicate the median score (expected reward), computed over 100 independently trained models. The filled area indicates 20–80 % confidence interval. **D.** We report average regret per round curves in OOD setting. We compare a gain-modulated network ($\gamma = 1$, teal), and a network without gain modulation ($\gamma = 0$, orchid) with the policy gradient training source (in black). Each curve represents the average regret per round for one fixed trained model, computed by averaging over 100 independent simulations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

error (MSE) between the target and estimated trajectories in closed-loop scenarios (see Fig. 1C, solid and dashed lines represent respectively, median and 20th/80th percentile range. Statistics is evaluated over 10 realizations of the experiment). In Fig. 1D we show the comparison of the learning curve for one $\omega = 0.055$, for the original learning algorithm, and the one evaluated by the two networks, in blue and orange respectively.

3.2. Extension to reinforcement learning and the role of gain modulation

Dynamical adaptation is now investigated within the framework of reinforcement learning. We provide robust evidence supporting the hypothesis that gain modulation represents a fundamental component in implementing in-context learning in biological agents. We first explore stateless environments $\mathcal{A}^{bandits}$, which are represented by Bernoulli K-armed bandits (Berry & Fristedt, 1985). Within each environment $\alpha \in \mathcal{A}^{bandits}$, there exists a subset $P_\alpha \subset [K]$ of arms that yield a reward with high probability ($p = .95$). During the training phase, the in-distribution environments give a high reward probability to even-numbered arms, whereas during testing, the out-of-distribution environments assign a high reward probability to odd-numbered arms.

In this simplified bandit scenario, where state information is absent, the virtual weights $\mathbf{w}$ directly parameterize the policy probabilities: $\boldsymbol{\pi} = \mathrm{softmax}(\mathbf{w})$. Consequently, our focus lies primarily on analyzing the behavior of the network $W^{\gamma}_{\Theta^w}(\cdot)$ acting on the virtual parameters (Fig. 2A, green). This serves as an ideal test bed for evaluating the network's ability to learn the policy gradient update rule and generalize it to out-of-distribution scenarios. The network $W^{\gamma}_{\Theta^w}(\boldsymbol{\pi}, a, r)$ has a gain-modulated architecture, as described in the Methods section, and is a function of the reward r and the action a. Here γ indicates the dependence on the gain-modulation strength. We train the network to approximate the policy gradient update rule using policy gradient estimates from a single learning trajectory (1000 rounds, learning rate $lr = 0.1$) in an in-distribution environment. We then test the distilled policy gradient networks with a higher learning rate ($lr = 1.0$) in out-of-distribution (OOD) environments. To systematically investigate the impact of gain modulation ($\gamma = 1$) on out-of-distribution performance, we compare it with networks where the reward does not modulate the network gain ($\gamma = 0$). For each case, we select the optimal hyperparameters through a grid-based search (additional details in the Appendix).

Our experimental findings are presented in Fig. 2. First, comparing models of different sizes (number N of hidden units), we find that models with gain modulation require significantly fewer neurons to achieve maximum scores in OOD environments (Fig. 2C). In Fig. 2B, we report the regret per round distribution at the 100th round and compare it with the distribution obtained by policy gradient with the same learning rate and iterations. A gain-modulated network achieves a regret distribution comparable to the policy gradient in both ID and OOD environments, with a lower median. In contrast, networks without gain modulation show significantly higher regret. Analyzing the regret curves in Fig. 2D, we observe that a model with gain modulation often learns a more data-efficient algorithm than its source, even in OOD environments. Conversely, a model without gain modulation fails to generalize to OOD environments and does not converge to zero regret.

In summary, gain modulation enables the network to consistently and efficiently distill the correct gradient update rule and generalize it to unseen environments, predicting the correct virtual weight update in regions of the input space far from the training data.

3.3. Dynamical reinforcement learning in a reaching task

We examine a reaching task, known as the dark room task (a simple instance of the water maze task (Morris, 1981)), set in a 2D maze within the domain $(-1, 1) \times (-1, 1)$ with a grid size of 0.1. Within this grid-like environment, the agent navigates by selecting one of four actions: up, down, left, or right, thereby determining its subsequent position. The primary goal is to locate a concealed object within the maze, with the agent having sole awareness of its own position. Feedback is provided via rewards, where the agent receives a reward of 10 if the distance to the object is 0.1, 15 if the distance is 0, and 0 otherwise. Through iterative exploration, the agent develops a strategy to efficiently traverse the maze and pinpoint the object despite the limited visibility. The position of the agent is encoded separately using 25 input units $\mathbf{x}$ each, employing Gaussian activation functions distributed on a 5x5 grid in the maze and with a width of 0.2.

To accomplish this task we consider a network architecture composed of two networks. One GM-network, $W^{\gamma}_{\Theta^w}(\{\mathbf{z}, r, a, \boldsymbol{\pi}\})$ is responsible for estimating the necessary update of virtual weights through policy gradient (Fig. 3A, green network), while another GM-network $Y^{\Gamma}_{\Theta^y}(\{\mathbf{z}, \mathbf{w}\})$ estimates policy reconfiguration (Fig. 3A, shown in orange) resulting from changes in virtual weights. Here, the parameters γ, Γ define the strength of the gain modulation, as explained in the Methods section. Their dynamics can be described by the following equation Eq. (13). Indeed, we compare the case with (Fig. 3A) and without (Fig. 3B) gain modulation. We refer to Supporting Information for further details on the training procedure.

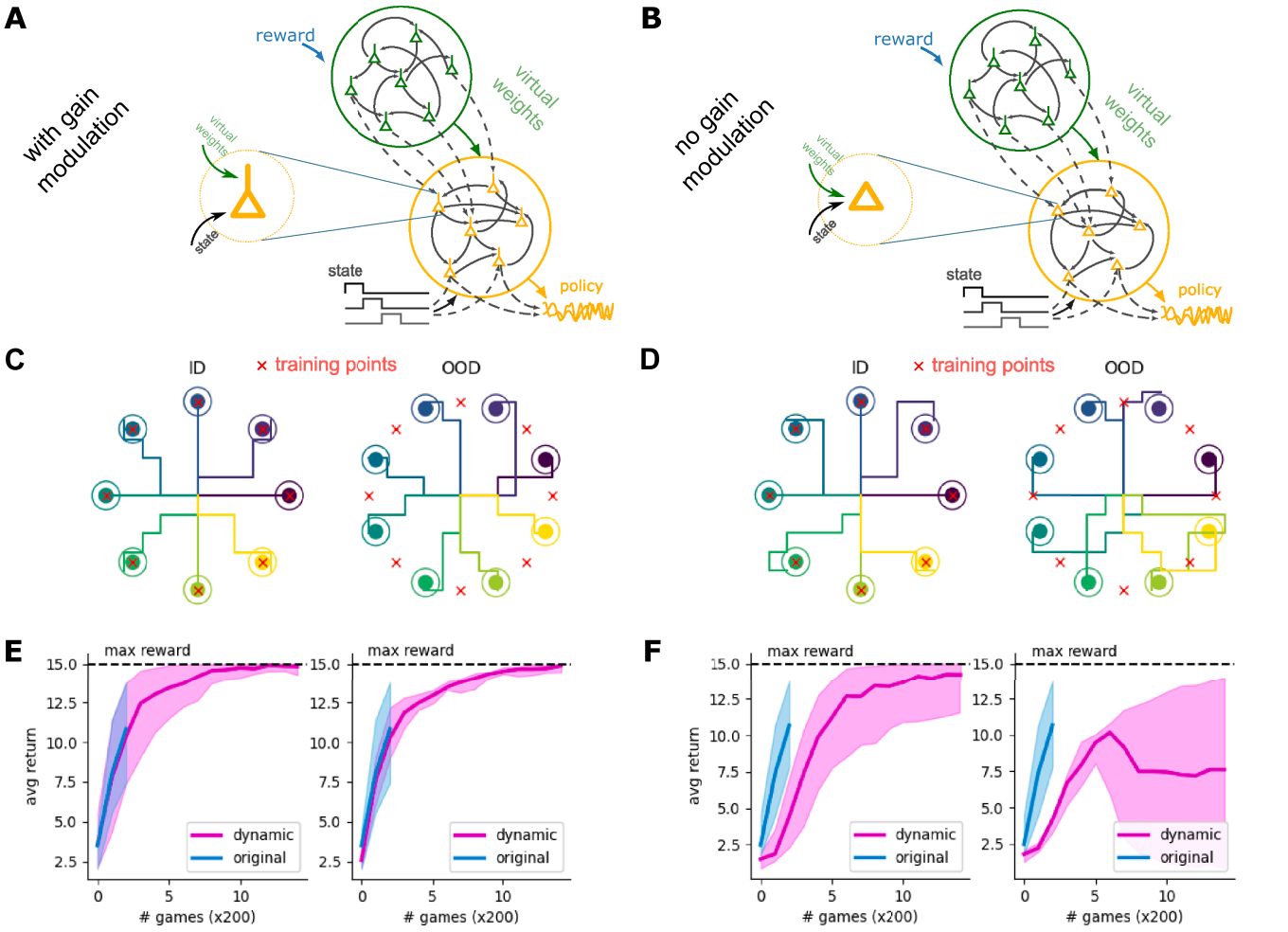

Fig. 3. Dynamical reinforcement learning dark room A. Overview of the network architecture employed. One GM-network (depicted in green) is responsible for estimating the necessary update of virtual weights through policy gradient, while another GM-reservoir is focused on estimating policy reconfiguration (shown in orange) resulting from changes in virtual weights. **C.** Illustration of the task: the agent begins from the center and learns to reach not observable objects positioned at various locations (represented by colored circles). After several trials, the agent achieves precise targeting of the circles. **C.** Reward as a function of number of trials (or games), blue: policy gradient, left panel pink: dynamical learning for training positions (ID), right panel pink: dynamical learning for new positions (OOD). Line: median, shading: 20-th/80-th percentile range. **B, D, F.** Similar to **A, C, E,** respectively, but without gain modulation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Firstly, we assessed the performance of the policy gradient algorithm using a set of 8 food locations (refer to Fig. 3C, indicated by red crosses), where the total reward averaged over 200 trials was observed against the number of trials (Fig. 3E, depicted by blue lines, thin lines for individual positions and thick lines for the average across all positions). After multiple trials, the agent successfully achieved precise targeting of the circles. Data collected from these experiments were utilized to train our networks to estimate gradients andscalar products, as defined in Eq. (13).

To validate that our trained model is capable of dynamically implementing policy gradient itself, we tested it on both the training set locations (ID, Fig. 3C, left panel) and new test positions (OOD, Fig. 3C, right panel). For each food position (coded with different colors), we illustrate a sample trajectory executed by the agent (in corresponding colors) to reach the target at the end of the training. The agent's precision closely matches that of the plastic policy gradient learning rule. We present the reward plotted against the number of trials for both training (Fig. 3E, right panel, pink line) and test (Fig. 3C, right panel, pink line) food locations.

We compared these performances against an architecture lacking gain modulation ($\gamma = 0$), observing worst performances (see Fig. 3B, D, F). Notably, while performances for ID food locations are acceptable (Fig. 3E, left panel, pink lines), those for OOD cases are extremely poor (Fig. 3E, right panel, pink lines). This observation, coupled with the results from the preceding section, suggests that gain modulation is a crucial component in facilitating the generalization of adaptive behavioral capabilities in recurrent networks.

Temporal credit assignment and Future Reward. We demonstrate, that our architecture is capable to perform the temporal computation required to learn delayed action-reward temporal relations (see Section D.1 in Appendix), requiring evaluating temporal credit assignment. Indeed, in reinforcement Learning (RL) an agent learns to maximize cumulative rewards. The discount factor, usually denoted by γ_d (where $0 \leq \gamma_d \leq 1$), is crucial in RL as it determines the importance of future rewards. A higher γ_d values future rewards more significantly, encouraging the agent to consider the long-term consequences of its actions. Conversely, a lower γ_d makes the agent prioritize immediate rewards. The cumulative reward R_t at time step t is given by:

$$R_t = \sum_{s \geq t} r_s \gamma_d^{(s-t)} \tag{3}$$

To account for this, the term $\mathbf{e}$ in Eq. (13) should be replaced by its exponential temporal filtering, with a timescale $\tau_e = -\frac{1}{\log(\gamma_d)}$ (see also

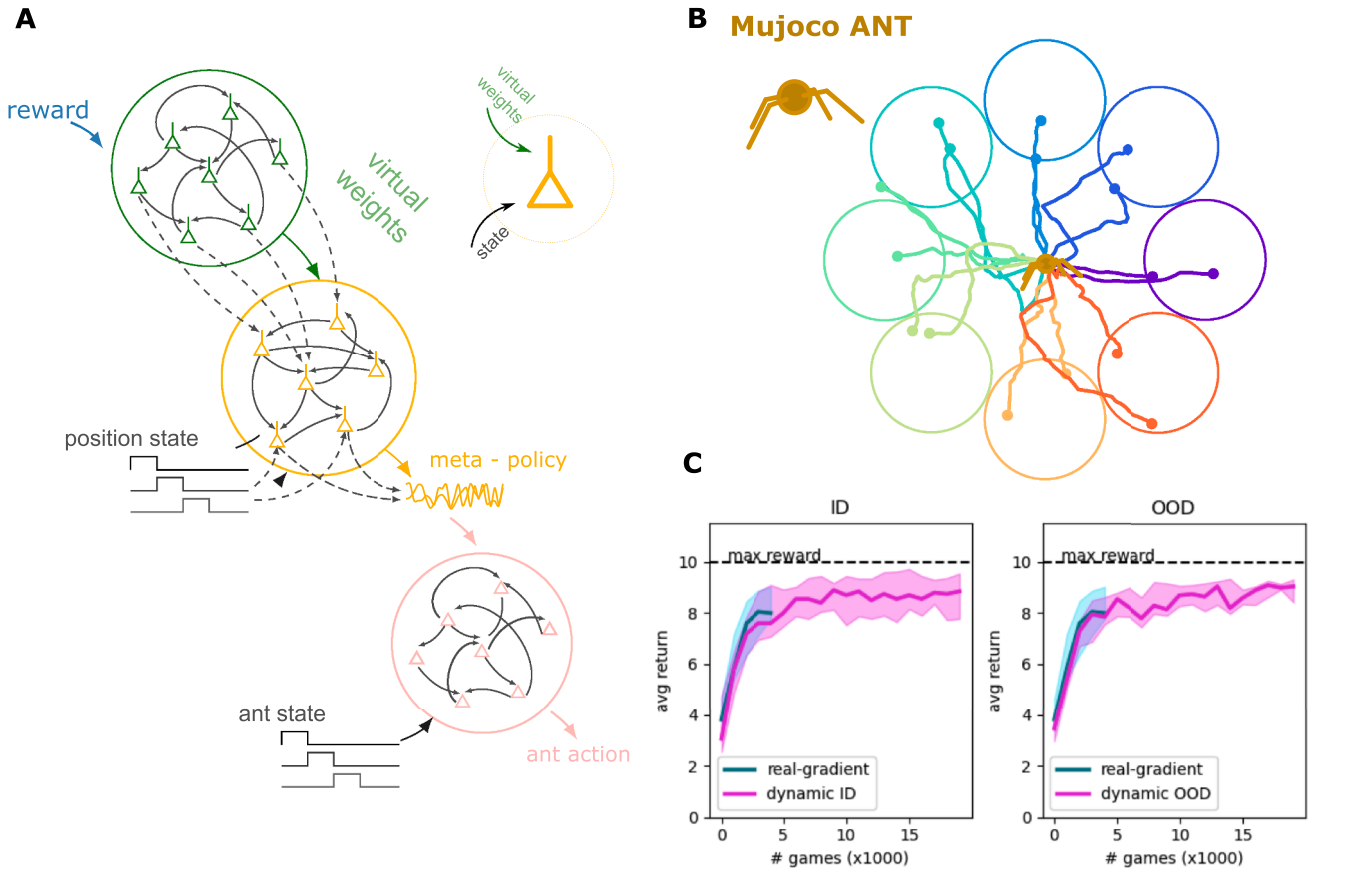

Fig. 4. Dynamical reinforcement learning dark room A. Overview of the network architecture employed. One GM-network (depicted in green) is responsible for estimating the necessary update of virtual weights through policy gradient, while another GM-reservoir is focused on estimating policy reconfiguration (shown in orange) resulting from changes in virtual weights. **B.** Illustration of the task, and of the MuJoCo environment: the agent begins from the center and learns to reach not observable objects positioned at various locations (represented by colored circles). After several trials, the agent achieves precise targeting of the circles. Our networks are pre-trained on a set of target points (red crosses) an then tested on the same (displayed in the left panel) and new positions (shown in the right panel). **C.** Reward as a function of number of trials (or games), blue: policy gradient, left panel pink: dynamical learning for training positions (ID), right panel pink: dynamical learning for new positions (OOD). Line: median, shading: 20-th/80-th percentile range. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Eq. (D.1) in the Appendix). In this way actions in the past, can be potentiated when delayed rewards are received.

If the discount factor is zero, the agent's policy would change only when it is very close to the moment the reward is received. In a reaching task, the agent will only change its policy when it is very close to the reward, completely ignoring the need for long-term planning and making it unlikely to ever reach the goal from distant starting points (see Fig. A.3D-E). As a result, the policy points towards the target position only when nearby the target itself (see Fig. A.3F), resulting in failure when the agent randomly moves in the wrong direction at the beginning of the task (see Fig. A.3F, black line).

On the other hand, recurrent connections enable a network to maintain a memory of past states and actions, effectively allowing it to use information from previous time steps to inform current decisions, achieving optimal planning and decisions even when far from the target location (and hence, from the reward, see Fig. A.3A–C).

Application to a robotic benchmark

We test our framework to a more complex and realistic setting by applying it to a MuJoCo ant environment, demonstrating its scalability to continuous control tasks. The agent, modeled as a quadruped, operates in a high-dimensional control space with 27 environmental state variables and 8 control torques (2 per leg), requiring coordinated control of multiple degrees of freedom.

To manage low-level motor execution, we first trained a network to perform four fundamental motor primitives–moving left, right, forward, and backward. This pretraining step allows higher-level adaptation without direct joint-level optimization (see Methods for details).

Building on this, we use our gain-modulated network as a meta-policy that dynamically selects motor primitives based on the agent's position. The meta-policy network (orange network Fig. 4A) receives positional inputs and determines the appropriate primitive, while a secondary network (green network Fig. 4A) modulates its behavior through virtual weights, mirroring the architecture used in the grid-based task.

The high-level policy is defined as a discrete policy over four options, corresponding to the four sub-policies. It is computed using a softmax over a linear readout of the observation vector:

$$\pi_{high}(a_{high} \mid s) = \mathrm{softmax}(W s) \tag{4}$$

where s is the current state of the environment, and $a_{high} \in \{\text{left}, \text{right}, \text{forward}, \text{backward}\}$. The selected action determines which sub-policy to invoke at each high-level decision point. As in the previous experiment, W and π_{high} are evaluated respectively by the networks $W^{\gamma}_{\Theta^w}$ and $Y^{\Gamma}_{\Theta^y}$.

This architecture enables flexible and efficient adaptation: the agent successfully generalizes to target locations not included in the training set (see Fig. 4C), adjusting its movement strategies dynamically without modifying its underlying synaptic structure. This capability is

particularly significant for neuromorphic computing and robotics, where adaptive behavior is often constrained by the need for on-chip plasticity. Our results demonstrate that a fixed neuromorphic chip, programmed with a gain-modulated network, can dynamically adjust to new goals without requiring structural modifications. This approach reduces computational overhead and power consumption while enabling efficient, low-power autonomous systems that adapt in real time. By eliminating the need for weight updates at the hardware level, this architecture provides a scalable and robust foundation for adaptive physical agents, paving the way for next-generation neuromorphic robotics.

4. Discussion

The brain's ability to flexibly and rapidly adapt to novel tasks remains only partially understood. Different studies have highlighted that this adaptability relies on mechanisms such as dynamic network reconfiguration and context-dependent modulation, allowing agents to respond flexibly to environmental changes. However, a significant gap remains in understanding how these processes map onto specific neural substrates.

While synaptic plasticity has been extensively studied as a mechanism underlying learning and memory (Abraham et al., 2019), emerging evidence suggests that it alone may not account for the full spectrum of adaptive behavior. Abraham et al. critically evaluate the synaptic plasticity and memory (SPM) hypothesis, concluding that non-synaptic mechanisms, such as intrinsic plasticity, structural neural changes, epigenetic regulation, glial contributions, network reorganization, and protein turnover, likely contribute significantly to memory storage and retrieval, thus broadening the framework for understanding dynamic adaptation. This perspective questions long-standing assumptions about neural circuits, highlighting that we may still lack full understanding of many processes we consider well-established.

Inspired by the brain's adaptability, artificial neural network architectures have been challenged to replicate similar capabilities. Fast synaptic plasticity, hypothesized to orchestrate flexible cognitive functions, inspired several artificial architectures, such as fast-weight networks (Irie et al., 2021; Schmidhuber, 1992) and hyper-networks that predict main network weights (Ba et al., 2016). However, these approaches often rely on biologically unrealistic mechanisms, such as requiring neurons to directly act on the synapses of a second network or implementing "inner loops" in network dynamics (Ba et al., 2016; Chalvidal et al., 2022). More biologically plausible alternatives, like coupled neural-synaptic dynamics (Miconi et al., 2018), come closer to natural systems but still depend on complex credit assignment mechanisms. Millidge et al. (2024) and Walter et al. (2024) address the problem of dynamic adaptation by proposing a temporal predictive coding model that uses recurrent connections to predict its own future neural responses and updates its parameters online through local plasticity. This allows the network to dynamically infer and track the state of continuously changing stimuli in real time. Complementary to those approaches, our method does not presuppose a specific learning paradigm or fixed synaptic plasticity rule. Instead, our framework demonstrates how a neural system can dynamically reproduce a target learning rule through activity-based mechanisms.

Transformers, known for their remarkable in-context learning (ICL) capabilities, excel at tasks requiring adaptability, such as natural language and image processing (Achiam et al., 2023; Ramesh et al., 2021; Vaswani et al., 2017). Mechanisms such as attention, associative memory (Ramsauer et al., 2020), and induction-based copying by attention heads (Olsson et al., 2022) have been proposed to explain ICL in transformers, with theoretical work showing that attention mechanisms can implement gradient updates (Von Oswald et al., 2023). However, the emergence of these capabilities remains poorly understood, particularly when comparing them to the in-context learning observed in biological systems.

We propose a constructive method to induce in-context learning in biologically plausible neural networks, in a broad variety of scenarios. We offer an alternative formulation in which dynamic adaptation is modulated by dendritic input segregation and amplification. Virtual (fast) weights are encoded by the activity of a second network that modulates the transfer function of neurons in the base network. As a result, virtual weights are not synaptic weights, but their values and updates are evaluated and encoded by the activity of other neurons, which were pre-trained to perform gradient updates. Another pre-trained network receives those weights signals as an input, along with the current environment features, and computes an adapted in-context response.

This mechanism overcomes the strict locality constraints of synaptic plasticity, which require all elements involved in evaluating weight updates to be present at the synaptic site. Consequently, biologically plausible synaptic updates cannot explicitly rely on information from other weights, a requirement often seen in backpropagation methods. In contrast, virtual weights are encoded by neuronal activity and can propagate across the neural circuit, bypassing these limitations. This allows virtual weight updates to depend on other virtual weights, enabling a more natural and precise evaluation of credit assignment.

Our work generalizes and provides a biologically plausible implementation for deep gated RNNs performing ICL (Zucchet et al., 2023), maintaining a similar number of trainable parameters. By dividing the architecture into two components and introducing an additional objective that forces one network to approximate the gradient of the other, we achieve robust dynamic adaptation in tasks requiring temporal learning. When tasked to learn temporal trajectories, through error feedback and virtual weight updates, our network achieves successful dynamical adaptation, without synaptic plasticity. Similarly, in reinforcement learning scenarios, the policy in response to the environment state is dynamically modulated by virtual weights, which are updated in function of the reward. We find that networks with gain modulation exhibit improved performance and robustness compared to those without gain modulation. Moreover, these gain-modulated networks demonstrate more data-efficient learning algorithms, outperforming counterparts in both in-distribution and out-of-distribution environments, showing their capability to face novel scenarios.

Our findings demonstrate that adaptive network architectures can emerge by selecting appropriate readout weights in initially random networks, without requiring explicit backpropagation. These weights can be shaped through biologically plausible mechanisms such as evolutionary processes, slow reinforcement learning, or local plasticity rules relying on feedback alignment. Unlike standard deep-learning models–such as transformers, gated RNNs, and hyper-networks–that depend on non-local credit assignment mechanisms like backpropagation through time, our approach leverages local, dynamically reconfigurable processes, making it more compatible with biological learning principles.

In line with this, our model suggests that dynamic reconfiguration provides a biologically plausible mechanism for rapid adaptation, complementing traditional fast synaptic plasticity. Different neural circuits in the brain may rely on varying balances of dynamic reconfiguration and synaptic plasticity, shaped by their functional roles and evolutionary pressures. In the Supplementary Materials, we provide a proof of concept through a T-maze simulated experiment, where we observe a substantial difference in how contextual information is encoded for synaptic versus adaptive learning. This distinction supports the potential for experimental validation.

In conclusion, our approach provides an explicit framework to induce in-context learning (ICL) in biologically plausible networks, offering a constructive pathway toward understanding ICL in biological systems and bridging the gap between neuroscience and machine learning.

5. Methods

5.1. Gain modulated reservoir computing (GM-RC)

Reservoir Computing. Reservoir Computing (RC) represents a paradigm in machine learning that provides a model for biologically plausible computation and learning, drawing inspiration from the information processing mechanisms observed in biological neural networks.

In this approach, a random RNN is employed to extract features from a time-dependent signal $\mathbf{x}(t)$. The dynamics of the RNN describe the evolution of N hidden units $\mathbf{z}(t) = (z_1(t), \cdots, z_N(t))$, governed by the following differential equation:

$$\tau_z \dot{\mathbf{z}} = \phi(J\mathbf{z} + R\mathbf{x}) - \mathbf{z} \tag{5}$$

The value of each unit represents the activity of a population of neurons following the Wilson and Cowan formulation (Wilson & Cowan, 1972). Here, J and R represent fixed random matrices, representing the recurrent connections and the projection from inputs to hidden units, respectively. Subsequently, the features extracted by the network can be utilized to predict a target signal $\mathbf{y}^{targ}(t)$ by learning readout weights Θ, such that $|\mathbf{y}^{targ}(t) - \Theta\mathbf{z}(t)|^2$ is minimized. The reservoir is then implementing a map

$$\mathbf{y}(t) = \mathrm{RNN}_\Theta(\{\mathbf{x}(s)\}_{s\le t}) \tag{6}$$

where Θ represents trainable parameters. In the rest of the article, we will write $\mathbf{y} = \mathrm{RNN}_\Theta(\{\mathbf{x}\})$, dropping the temporal dependencies. [1]

When addressing input-output mappings without time dependencies, we consider a network operating in the $\tau_z, J_{ij} \to 0$ limit. In this scenario, the network computes an instantaneous function of the input, represented as $\mathrm{NN}_\Theta(x) = \Theta\phi(R\mathbf{x})$. Essentially, this is equivalent to considering a one-layer feed-forward network with random fixed input weights. This architecture is also referred to in the literature as the Extreme Learning Machine (Huang et al., 2006).

Gain modulated network architecture. Analysing Eqs. (9) and (13) we observe that a network must possess the capability to perform multiplications of its inputs to approximate a gradient descent update of virtual parameters. Building upon this insight, we introduce a gain-modulated reservoir network (GM-RC). This architecture draws inspiration from the morphology and function of pyramidal neurons in the cortex, which nonlinearly integrate inputs from basal and apical dendrites (Larkum, 2013b; Shai et al., 2015). Here, we consider an additional input source $\mathbf{x}^{\mathrm{ap}}$ randomly projected into the apical dendrite of each neuron by the matrix R^{ap}. Consistent with experimental observations in L5 pyramidal neurons, we allow the apical inputs to modulate the gain of the activation function, thereby altering its slope. Consequently, the resulting RNN equation is formulated as:

$$\tau_z \dot{\mathbf{z}} = \phi\Big(\big(\mathbf{b}^{\mathrm{ap}} + \gamma \cdot R^{\mathrm{ap}}\mathbf{x}^{\mathrm{ap}}\big) \odot (J\mathbf{z} + \beta \cdot R^{\mathrm{ap}}\mathbf{x}^{\mathrm{ap}} + R\mathbf{x})\Big) - \mathbf{z} \tag{7}$$

Where $\mathbf{b}^{\mathrm{ap}}$ is a constant bias vector. The hyperparameters β, γ modulate the effect of the gain modulation of the apical inputs. Specifically, when $\gamma = 0$, we obtain a network in which $\mathbf{x}^{\mathrm{ap}}$ does not affect the gain modulation. Similarly, as before, the expression $\mathbf{y} = \mathrm{GMRNN}^\gamma_\Theta(\{\mathbf{x}|\mathbf{x}^{\mathrm{ap}}\})$ will denote the input → output mapping implemented by a gain-modulated reservoir. As before, removing time dependencies, we will have an instantaneous function $\mathbf{y} = \mathrm{GMNN}^\gamma_\Theta(\mathbf{x}|\mathbf{x}^{\mathrm{ap}}) = \Theta\phi((\mathbf{b}^{\mathrm{ap}} + \gamma \cdot R^{\mathrm{ap}}\mathbf{x}^{\mathrm{ap}}) \odot (R\mathbf{x}))$ of the inputs $\mathbf{x}^{\mathrm{ap}}$ and $\mathbf{x}$. We explicitly maintain the dependence on γ because, in our experiments, we use this parameter to regulate the gain modulation effect of the apical inputs on the network. The choice for the name $\mathbf{x}^{\mathrm{ap}}$ is inspired by the current received in the apical dendrites of L5 pyramidal neurons, that are believed to carry contextual/high-level information (Larkum, 2013b).

[1] For this mapping to be well-defined, we can assume the RNN to have the echo-state property (Yildiz et al., 2012).

5.2. Dynamical supervised learning for temporal trajectory

We consider the task of learning a target temporal trajectory. Each environment α is defined by a target trajectory $y^{targ}_\alpha(t)$, such that the input and the feedback $x_\alpha(t) = f_\alpha(t) = y^{targ}_\alpha(t)$ are both given by the target sequence. The network response is defined as the current estimation of $y^{targ}(t)$ via a linear readout $y(t) = \mathbf{w} \cdot \mathbf{z}(t)$, where $\mathbf{z}$ are the activity of a random RNN that operates as a reservoir computer extracting nonlinear features of the input sequence. The RNN dynamics is given by Eq. (5), with input x given by the target sequence. The readout weights in turn dynamically adapt by following the delta rule, minimizing the reconstruction error.

The learning process can then be described by the following dynamical system:

$$\begin{cases} e & = (y^{targ} - y) \\ y & = \mathbf{z} \cdot \mathbf{w} \\ \tau_z \dot{\mathbf{z}} & = \phi(J\mathbf{z} + R\mathbf{x}) - \mathbf{z} \\ \tau_w \dot{\mathbf{w}} & = \mathbf{z}\, e \end{cases} \tag{8}$$

where we removed the dependence on time t for simplicity. The operations required are nonlinear, and usually are viewed as implemented by the plasticity rule and by the multiplication between presynaptic activity and synaptic weights. However, here we interpret $\mathbf{w}$ as dynamic variables, proposing, as a possible implementation, that such non-linear functions are computed by two neural networks W_{Θ^w}, and Y_{Θ^y}:

$$\begin{cases} e & = (y^{targ} - y) \\ \tau_z \dot{\mathbf{z}} & = \phi(J\mathbf{z} + R\mathbf{x}) - \mathbf{z} \\ y & = Y_{\Theta^y}(\{\mathbf{x}|\mathbf{w}\}) \simeq Y(\mathbf{x}, \mathbf{w}) = \mathbf{x} \cdot \mathbf{w} \\ \tau_w \dot{\mathbf{w}} & = W_{\Theta^w}(\{\mathbf{x}|e\}) \simeq W(\mathbf{x}, e) = \mathbf{x}\, e \end{cases} \tag{9}$$

In particular, we used two GMRNNs that receive $\mathbf{x}, \mathbf{w}, e$ as inputs and provide the proper output thanks to a suited training of their readout weights (Θ^y and Θ^w, following reservoir computing paradigm). Notice that now Eq. (9) represents an RNN dynamically performing the learning process.

5.3. Approximation properties of bilinear networks

To better understand the expressive power of gain modulated networks we study the approximation properties of feedforward networks with linear transfer functions ($\phi = Id$).

In this case, we have a bilinear layer, such that it naturally parameterizes a family of linear functions, indexed by the virtual weights:

Proposition 1. *Let $Y(\{\mathbf{x}|\mathbf{w}\}) = \Theta((R^{qp}\mathbf{x}^{qp}) \odot (R\mathbf{x}))$ where Θ, R^{ap}, and R are random matrices of dimensions $1 \times N_h$, $N_h \times N_{in}$, and $N_h \times N_{in}$, respectively, with i.i.d. entries drawn from any distribution that is absolutely continuous with respect to the Lebesgue measure.* [2] *Then almost surely, if $N_h \ge N_{in}$ for every linear function, $f : \mathbb{R}^{N_{in}} \to \mathbb{R}$ there exists a set of virtual parameters $\mathbf{w}_f \in \mathbb{R}^{N_{in}}$ such that $Y(\{\cdot|\mathbf{w}_f\}) = f(\cdot)$.*

Proof. We first observe that the output can be written as a bilinear form

$$Y(\{\mathbf{x}|\mathbf{w}\}) = \mathbf{w}^\top A\mathbf{x} \tag{10}$$

where the matrix $A \in \mathbb{R}^{N_{\mathrm{in}} \times N_{\mathrm{in}}}$ has entries

$$A_{ij} = \sum_{k=1}^{N_h} \Theta_{1k} R^{\mathrm{ap}}_{ki} R_{kj} \tag{11}$$

That is, A is a sum of N_h rank-one matrices of the form $\Theta_{1k}\mathbf{R}^{\mathrm{ap}}_k \mathbf{R}^\top_k$, where $\mathbf{R}^{\mathrm{ap}}_k$ and $\mathbf{R}_k$ are the k-th rows of R^{ap} and R, respectively.

Since the entries of Θ, R^{ap}, and R are i.i.d. and drawn from a distribution absolutely continuous with respect to the Lebesgue measure, the

[2] Here for simplicity we assumed $\gamma = 1$, $\beta = 0$, $\mathbf{b}^{\mathrm{ap}} = 0$.

collection of vectors $\{R_k^{\text{ap}}\}_{k=1}^{N_h}$ $\{R_k\}_{k=1}^{N_h}$ almost surely span a subspace of dimension $\min(N_h, N\text{in})$. Consequently, the matrix A has rank $rank(A) = \max(N_h, N_{in})$ almost surely. Thus, if $N_h \geq N_{\text{in}}$, then A is almost surely invertible. Now, given any linear function $f(\mathbf{x}) = \mathbf{w}_*^\top \mathbf{x}$ with $\mathbf{w}_* \in \mathbb{R}^{N_{\text{in}}}$, define $\mathbf{w}_f := (A^\top)^{-1}\mathbf{w}_*$. Then, for all $\mathbf{x}$, $Y(\{\mathbf{x}|\mathbf{w}_f\}) = \mathbf{w}_*^\top \mathbf{x} = f(x)$ as required. □

Consider a gain-modulated bilinear network that dynamically approximates a continuous in-context target signal $\hat{y}(t)$ by adjusting the activity of its virtual weights to minimize the mean squared error $(Y(\{\mathbf{x}|\mathbf{w}\}) - \hat{y}(t))^2$. The following proposition shows that a separate gain-modulated bilinear network can express the corresponding online gradient update using a number of units that scales linearly with the number of features.

Proposition 2. *Let $Y(\{\mathbf{z}|\mathbf{w}\})$ defined as in Proposition 1, and $W_{\Theta^w}(\{\mathbf{z}|e\}) = \Theta^w((R^{ap}e) \odot (R\mathbf{z}))$ another gain-modulated bilinear network with R^{ap}, R are random matrices of dimensions $1 \times N_h$, and $N_h \times N_{in}$, respectively, with i.i.d. entries drawn from any distribution that is absolutely continuous with respect to the Lebesgue measure. Then there exists a readout matrix Θ^w such that $W_{\Theta^w}(\{\mathbf{z}|e\}) = e\mathbf{z} = e\nabla_{\mathbf{w}} Y(\{\mathbf{z}|\mathbf{w}\})$*

Proof. We can express that the output can be written as $W_{\Theta^w}(\{\mathbf{z}|e\}) = e\Theta^w C\mathbf{z}$ where $C_{ij} = R_{1j}^{\text{ap}} R_{ij}$. Since the entries of R^{ap}, and R are i.i.d. and drawn from a distribution absolutely continuous with respect to the Lebesgue measure, when $N_h \geq N_{in}$ the matrix C is almost surely injective. Then there exists a left inverse Θ_* such that $\Theta_* C = Id$. □

5.4. Dynamical reinforcement learning

In the context of reinforcement learning, our framework outlines a systematic approach for modelling agents that can dynamically adapt their behavior across diverse environments. We assume the same state transition dynamics across all the environments $\dot{\mathbf{x}} = X(\mathbf{x}, a)$, while the reward distribution $r \sim Prob_\alpha(r|\mathbf{x}, a)$ depends on the current environment $\alpha \in \mathcal{A}$. Here $a(t)$ is an integer value indicating the action at time t among D possible ones.

We start by defining a policy network, denoted as $\boldsymbol{\pi} = \text{softmax}(\mathbf{y})$ which implements a policy mapping the agent state encoded by the feature vector $\mathbf{z}$, to a probability distribution over actions. We assume an agent that linearly recombines these features $\mathbf{y} = Y(\mathbf{z}, \mathbf{w}) = \mathbf{w} \cdot \mathbf{z}$. This assumption is made without loss of generality, since we could always employ a reservoir computer as an intermediate feature extraction layer, as described in the previous subsection. The agent learns by evaluating the policy gradient with respect to the weights $\mathbf{w}$. This leads to a dynamics similar to Eq. (9), where the definition of $e(t)$ is changed as follows:

$$\begin{cases} \mathbf{e} & = r\big(\mathbb{1}_a - \boldsymbol{\pi}\big) \\ \mathbf{y} & = \mathbf{w} \cdot \mathbf{z} \\ \tau_w \dot{\mathbf{w}} & = \mathbf{e} \odot \mathbf{z} \end{cases} \tag{12}$$

Here $\mathbb{1}_{a(t)}$ represents the 'one-hot encoded' action (as defined in Bellec et al., 2020; Capone & Paolucci, 2022) at time t. It is a D-element vector where the $a(t)$-th element is one, and all other elements are zero. This formulation holds for a null discount factor (Bellec et al., 2020; Capone & Paolucci, 2022), we refer to the Appendix Section D.1 for the description of the general case.

As before, we model that the policy is modulated by an auxiliary population of virtual weights $\mathbf{w}$, determined by the activity of an additional GMRNN W_{Θ_y}. This auxiliary network adjusts its internal activity based on the rewards received, effectively implementing policy gradient updates of the virtual weights and thereby modulating the agent behavior in real-time. Then, the above dynamics can be approximated by two networks, one for estimating thegradients, and one for the scalar product.

$$\begin{cases} \tau_w \dot{\mathbf{w}} & = W_{\Theta^w}(\{\mathbf{z}|r, a, \boldsymbol{\pi}\}) \simeq W(\mathbf{z}, r, a, \boldsymbol{\pi}) = \\ & = r\big(\mathbb{1}_a - \boldsymbol{\pi}(\mathbf{y})\big) \odot \mathbf{z} \\ \mathbf{y} & = Y_{\Theta^y}(\{\mathbf{z}|\mathbf{w}\}) \simeq Y(\mathbf{z}, \mathbf{w}) = \mathbf{x} \cdot \mathbf{w} \end{cases} \tag{13}$$

5.5. Training the low-level network for Mujoco-ANT

We conducted a reinforcement learning experiment using the MuJoCo-based `Ant-v4` environment in OpenAI Gym. The agent's objective was to learn movement patterns through a graded goal-directed reinforcement learning model. The experiment was run for 500,000 episodes, with each episode lasting 400 timesteps. The agent utilized a neural network model with 500 recurrent units, an input size of 31 (27 state variables + 4 target indicators, corresponding to UP, DOWN, LEFT, and RIGHT), and an output dimension of 8 (corresponding to the Ant's actuators). Each episode started with environment reset and the agent processing observations. The agent selected actions based on its internal policy, executed them, and received a reward. The reward was defined based on directional movement accuracy. Only readout weights where trained, following a policy gradient approach. Optimization was performed using the Adam optimizer.

CRediT authorship contribution statement

Cristiano Capone: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Methodology, Formal analysis, Data curation, Conceptualization; **Luca Falorsi:** Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Formal analysis, Conceptualization.

Source code availability

The source code is available under CC-BY license in the public repository https://github.com/cristianocapone/ABSS.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT in order to check sentence structure and grammar. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research has received financial support from the Italian National Recovery and Resilience Plan (PNRR), M4C2, funded by the European Union - NextGenerationEU (Project IR0000011, CUP B51E22000150006, 'EBRAINS-Italy'). LF aknowledges support by ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing and Sapienza University of Rome (AR12419078A2D6F9). Work partially funded by the European Union – NextGenerationEU and the Ministry of University and Research (MUR), Italian National Recovery and Resilience Plan (PNRR), M4C2I1.3, project 'MNESYS' (PE00000006). We sincerely thank Maurizio Mattia for helpful discussions on the topic.

Appendix A. Approximating gradients with gain modulated architectures and relationship with previous work

The relationship between attention-based transformer architectures and in-context learning was first noted in Von Oswald et al. (2023) where it was shown through constructive proof, confirmed by experiments, that linear attention layers can implement a gradient descent update for linear regression in its forward pass. Building upon this insight, subsequent work (Zucchet et al., 2023) showed that a linear gated RNN can implement the same mechanism, showing that a linear two-layer gated RNN can replicate a linear transformer. The work proposes an implementation that uses $O(d^2)$ hidden units and has $O(d^4)$ trainable weights, that can be reduced to $O(d^3)$ by side gating. Our gain-modulated architecture presents a generalized version of the linear gated RNNs used in the deep learning literature (Zucchet et al., 2023). To see this, consider two linear [3] GMNNs implementing the functions $Y_{\Theta^y}(\mathbf{z}, \mathbf{w})$, $W_{\Theta^w}(\mathbf{z}, y^{targ} - y)$, dynamically performing a ICL linear regression task that requires to map temporal dependent features $z(t) \in \mathbb{R}^{N_{in}}$ to predict an output $\mathbf{y}^{targ} \in \mathbb{R}^{N_{out}}$:

$$\begin{cases} \mathbf{y} & = Y_{\Theta^y}(\mathbf{z}, \mathbf{w}) = \Theta^y \left(R_y^{ap} \mathbf{w} \right) \odot (R_y \mathbf{z}(t)) \\ \tau_w \, \dot{\mathbf{w}} & = W_{\Theta^w}(\mathbf{z}, \mathbf{y}^{targ} - \mathbf{y}) = \Theta^w \left(R_w^{ap} (\hat{y} - y) \right) \odot (R_w \mathbf{z}(t)) \end{cases} \tag{A.1}$$

This architecture is the same as considered in the theoretical analysis in the methods section, and can be seen as a particular instance of the RNN with side gating (see Eq.(15) in Zucchet et al., 2023). Interestingly, a similar mechanism is also present in the recently proposed Mamba layer (Gu & Dao, 2023).

In the rest of this appendix, we will demonstrate that an architecture with gain modulation is better suited to approximate the gradient terms involved in dynamical learning within a simple univariate linear regression task on the features $\mathbf{z}(t)$. In this task, as described in the Methods section, the functions Y and W are represented by a dot product and a scalar-vector product, respectively. To simplify the analysis, we employ two GMNNs (gain-modulated neural networks), modeled as extreme learning machines (ELMs), without recurrent connections to approximate these functions. While an architecture without recurrent connections suffices for this task, more complex scenarios involving temporal credit assignment require recurrent connections, as shown in Fig. A.3. Notably, in the mathematical literature, the general approximation capability of an Echo State Network (ESN) can be rigorously understood through its relationship to an associated extreme learning machine (Hart et al., 2020).

Scalar product. We first consider the task of approximating a scalar product between virtual weights $\mathbf{w} \in \mathbb{R}^{N_{in}}$ and features $\mathbf{z} \in \mathbb{R}^{N_{in}}$. To achieve this, we train the readout weights Θ of a gain modulated network $\text{GMNN}^{\gamma}_{\Theta}(\mathbf{z}|\mathbf{w})$ with N_h hidden features to approximate the function $dot(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{N_{in}} z_i w_i$.

We use a training dataset composed of 1000 pairs $(\mathbf{z}, \mathbf{w})$ uniformly sampled in the hypercube $[0,1]^{2N_{in}}$. The test set consists of the same number of pairs sampled in the hypercube $[-1,1]^{2N_{in}}$. We compare an architecture with gain modulation ($\gamma = 1$) with an architecture without gain modulation $\gamma = 0$. For each of the two settings, we select the best hyperparameters by fixing the dimensionality of the inputs ($N_{in} = 5$) and hidden units ($N_h = 101$) and varying the standard deviation of the projection matrices $R_{ij}, R^{ap}_{ij} \sim \mathcal{N}(0, \sigma_R^2)$, $\log_{10}(\sigma_R) \in \{-2, -1.8, \cdots, 0\}$, the standard deviation of the bias $b_i \sim \mathcal{N}(0, \sigma_b^2)$, $\sigma_b \in \{0, 0.1, \cdots, 1\}$ and the nonlinearity type $\phi \in \{\tanh, \text{softplus}\}$. The best hyperparameters found were used in the models to test the approximation performance of the dot product varying the number of input features ($N_{in} \in [1, 10] \cap \mathbb{N}$) and hidden units ($N_h \in [1, 200] \cap \mathbb{N}$). Results are shown in Fig. A.1 A–C. We see that while both architectures require a quadratic number of hidden units to achieve low error on the test set, the gain-modulated network after a threshold number of units is hidden units is reached, is able to perfectly approximate the scalar product function consistently achieving $\sim 10^{-7}$ RMSE error. At the same time, the error in the network without gain modulation remains several orders of magnitude higher.

Scalar-vector product. We then tackle the task of approximating a vector-scalar product between features $\mathbf{x} \in \mathbb{R}^{N_{in}}$ and features and error $\mathbf{e} \in \mathbb{R}$. To achieve this, we train the readout weights Θ of a gain modulated network $\text{GMNN}^{\gamma}_{\Theta}(\mathbf{x}|e)$ with N_h hidden features to approximate the function $prod(\mathbf{x}, e) = e\mathbf{x} \in \mathbb{R}^{N_{in}}$.

We used a training dataset is composed of 1000 pairs $(\mathbf{x}, e)$ uniformly sampled in the hypercube $[0,1]^{N_{in}} \times [0,1]$. The test set consists of the same number of pairs sampled in the hypercube $[-1,1]^{N_{in}} \times [-1,1]$.

We compare an architecture with gain modulation ($\gamma = 1$) with an architecture without gain modulation $\gamma = 0$. For each of the two settings, we select the best hyperparameters by fixing the dimensionality of the inputs ($N_{in} = 5$) and hidden units ($N_h = 21$) and varying the standard deviation of the projection matrices $R_{ij} \sim \mathcal{N}(0, \sigma_R^2/N_{in})$, $R^{ap}_{ij} \sim \mathcal{N}(0, \sigma_R^2)$, $\log_{10}(\sigma_R) \in \{-2, -1.8, \cdots, 0\}$, the standard deviation of the bias $b_i \sim \mathcal{N}(0, \sigma_b^2)$, $\sigma_b \in \{0, 0.1, \cdots, 1\}$ and the nonlinearity type $\phi \in \{\tanh, \text{softplus}\}$. The best hyperparameters found were used in the models to test the approximation performance of the dot product varying the number of input features ($N_{in} \in [1, 10] \cap \mathbb{N}$) and hidden units ($N_h \in [1, 40] \cap \mathbb{N}$). Results are shown in Fig. A.1 D–F. We see that the network without gain modulation requires a quadratic number of hidden units to achieve low error on the test set. As in the scalar product case, the gain-modulated network can nearly perfectly approximate the target function after a threshold number of hidden units is reached. Significantly, in this case, the threshold scales linearly with $\mathbb{R}^{N_{in}}$.

On the number of hidden units and trainable weights needed in our model. The empirically found scaling can be easily understood assuming the gain-modulated network operates in a near linear regime (which here is sufficient, since we are learning a linear relationship between the features and the readout, this is confirmed by the low input variance found by the hyperparameter search). Considering a first-order Taylor expansion, the hidden units extract features that are (random) linear combinations of products of apical and basal inputs. Since the target function (both for Y and W) is also a linear combination products of apical and basal inputs, it is sufficient for the readout to linearly recombine these terms into the right form, as shown in Proposition 2.

In general, for gain-modulated architectures, we experimentally found a linear scaling for the W network and a quadratic scaling for Y (as a function of the number N_{in} of independent inputs). The scaling for Y can be further reduced to linear by tuning the apical R^{ap} projection matrix or by transforming the apical input $\mathbf{w}$ as proven in Proposition 1. This suggests that achieving efficient scaling may require mechanisms beyond standard algorithm distillation. Future research will focus on identifying biologically plausible processes that enable such adaptive weight adjustments.

Networks without gain modulation have significantly worse performance. To have products appear in the Taylor expansion for networks without gain modulation, we need a second-order Taylor expansion which produces a quadratic number of terms involving products between all input variables. This explains the quadratic scaling found for architectures without gain modulation in panels A and D of Fig. A.1.

Consider now the architecture in Eq. (A.1) performing an online univariate linear regression task ($N_{out} = 1$):

$$\begin{cases} y & = Y_{\Theta^y}(\mathbf{z}, \mathbf{w}) = \Theta^y \left(R_y^{ap} \mathbf{w} \right) \odot (R_y \mathbf{z}(t)) \\ \tau_w \, \dot{\mathbf{w}} & = W_{\Theta^w}(\mathbf{z}, y^{targ} - y) = \Theta^w \left(R_w^{ap} (\hat{y} - y) \right) \odot (R_w \mathbf{z}(t)) \end{cases} \tag{A.2}$$

Where $R_y^{ap}, R_y, R_w^{ap}, R_w$ are fixed random matrices respectively of dimension $N_y \times N_{in}, N_y \times N_{in}, N_w \times 1, N_w \times N_{in}$, and Θ^y, Θ^w are

[3] For simplicity, here we assume $\mathbf{b}^{ap}, \beta = 0$, $\gamma = 1$ and consider a linear activation function $\phi = Id$.

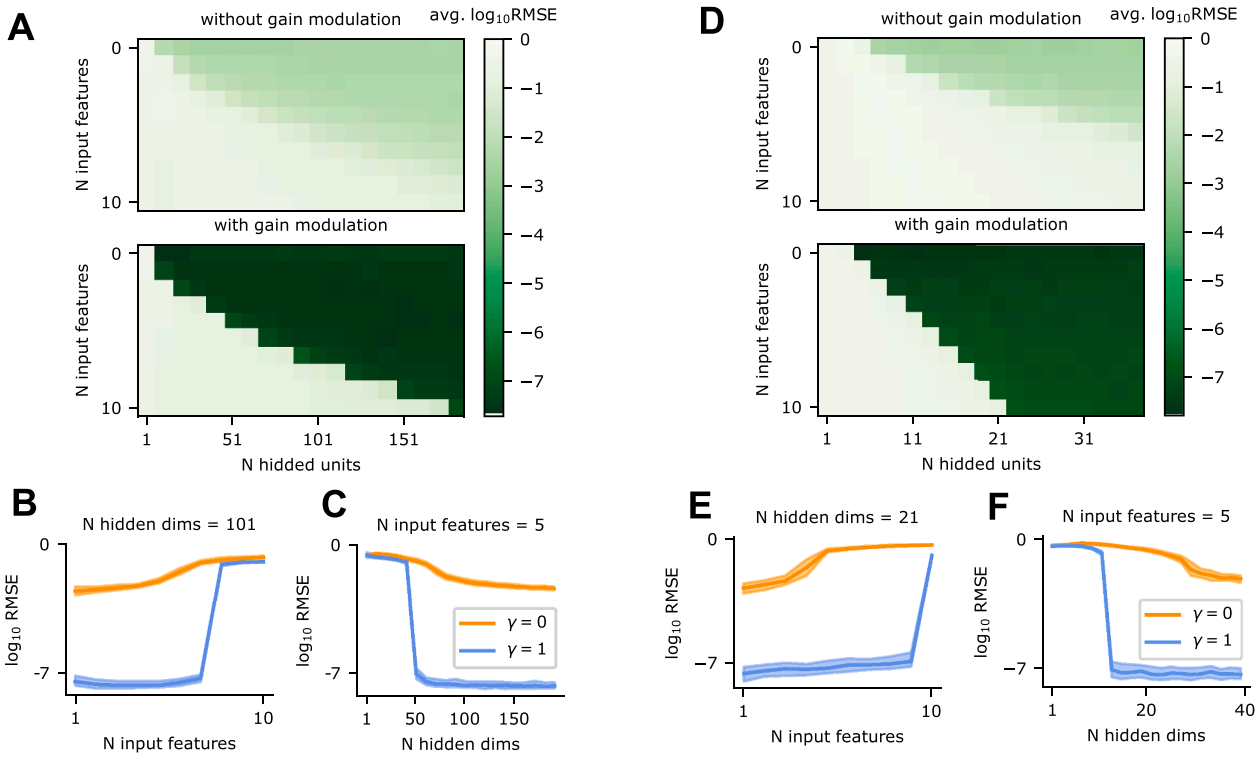

Fig. A.1. **A** Test errors for the scalar product approximation task, varying the number of features N_{in}, hidden units N_h. We compare architectures with gain modulation ($\gamma = 1$, bottom) with architectures without gain modulation ($\gamma = 0$, bottom). **B** Test errors for the dot product approximation task, varying the number of features N_{in}, and fixing the number of hidden units $N_h = 101$. Solid lines indicate the median over 100 trained models, while the filled region indicates the $20/80th$-percentile interval. **C** Same as **B** but fixing the number of features $N_{in} = 5$, and varying the number of hidden units $N_h = 101$. **D, E, F** Same as panels **A, B, C** but for the scalar-vector product approximation task.

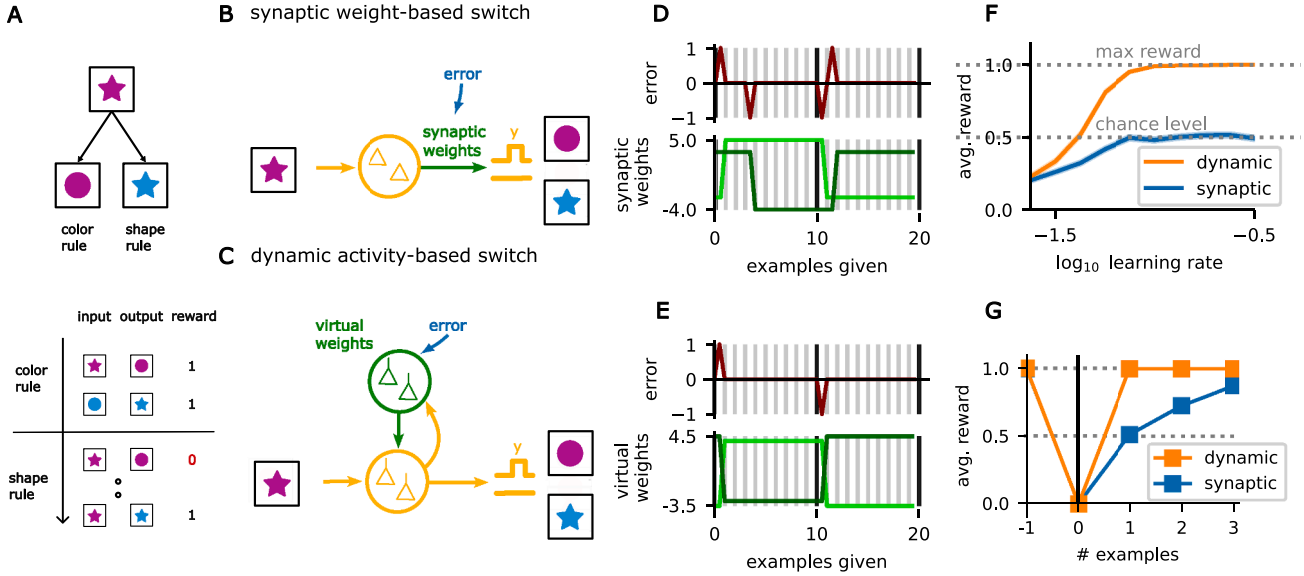

Fig. A.2. **Adapting behavior in a context-dependent classification task. A.** Overview of the task structure: At each iteration, the agent is presented with a cue card and two guide cards, one matching the cue by color and the other by shape. The agent must select the correct guide card to receive a positive reward. The classification rule switches after 20 trials. **B.** Schematic representation of the proposed network architecture: synaptic plasticity of physical weights is replaced by dynamic adaptation of virtual weights **C.** that modulate the network response. **D.** Example temporal trajectory for the plastic network. Two in-context errors are needed to learn the correct rule. **E.** Example temporal trajectory for the network with dynamic adaptation: one in-context errors are needed to learn the correct rule. **F.** Average reward after 1 in-context example as a function of the learning rate η. Synaptic plasticity with a local delta rule does not go beyond the chance level. Dynamic adaptation with a non-local learning rule can achieve a perfect score. The curve is obtained by averaging over 10 independent sessions of 2000 iterations each. The shaded region represents the standard deviation confidence interval. **G.** Psychometric curve comparing average reward as a function of in-context examples. At -1 we report the average reward before the rule switch. The curve is obtained by averaging over 10 independent sessions of 2000 iterations each.

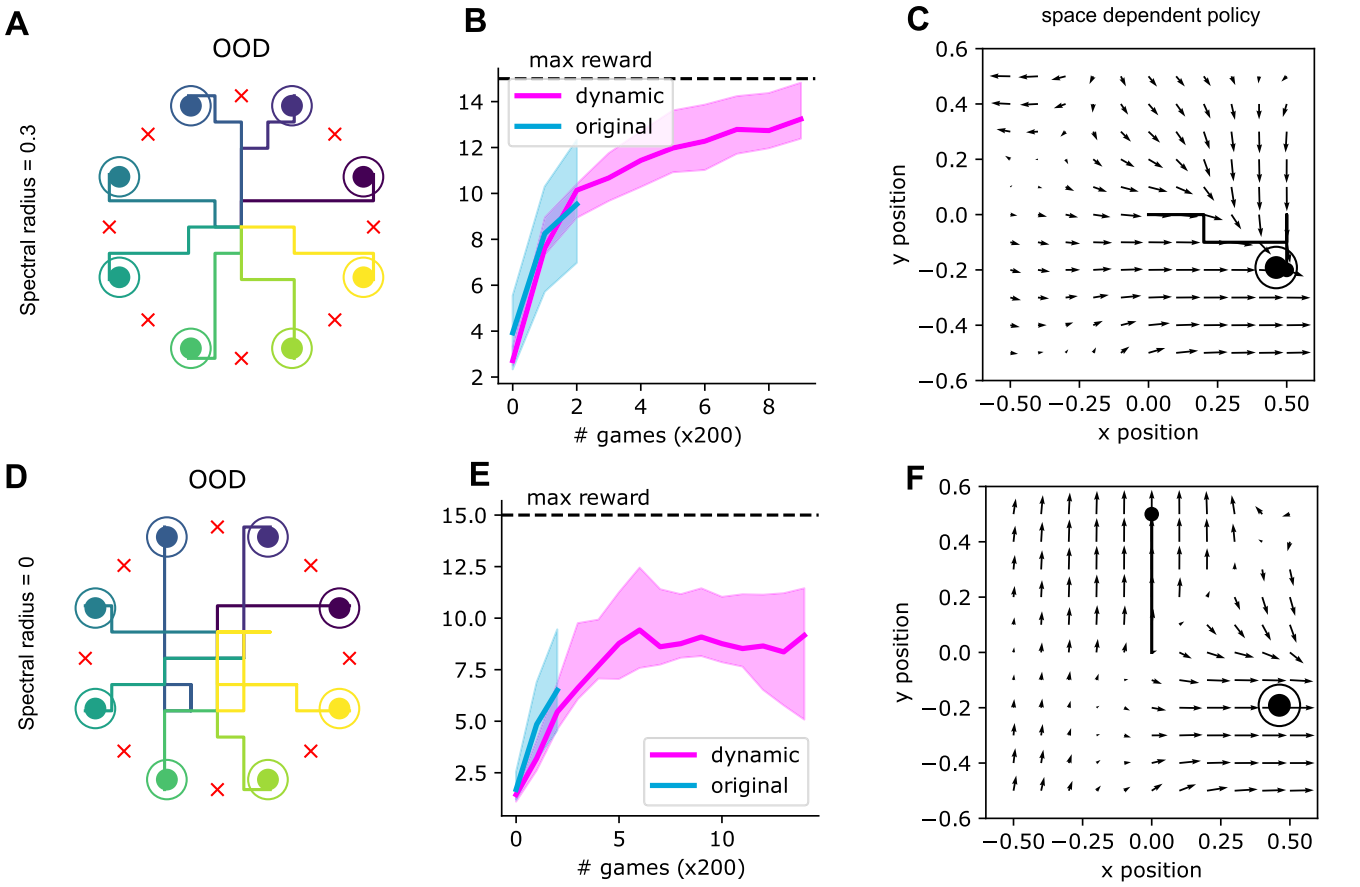

Fig. A.3. Discount factor and eligibility traces: A. Sample trajectories of the agent at the end of the learning procedure, for various target locations (color-coded), different from the ones used to train the gradient and the scalar product networks (red crosses). **B.** Reward as a function of number of trials (or games), blue: policy gradient, pink: dynamical learning for new positions (OOD). Line: median, shading: 20-th/80-th percentile range. **C.** Arrows: policy at the end of the training as a function of the position. Line: sample trajectory of the agent at the end of learning. Small circle: agent position, double circle: target position. **D-E-F** same as in **A-B-C**, but the recurrent connections of the gradient network are set to zero. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

trainable readout matrices of sizes $1 \times N_y, N_{in} \times N_w$. From the previous experiment and theoretical analysis, we need $N_w \sim O(N_{in}^2)$ hidden units and $O(N_{in}^2)$ readout parameters for the Y network. This number can be further reduced to $O(N_{in})$ both for the hidden units and trainable weights [4]. As for the network $W_{\Theta^W}(\mathbf{z}, y^{targ} - y)$ approximating vector-scalar products, $N_y \sim O(N_{in})$ hidden units and $O(N_{in}^2)$ readout parameters are needed.

For a multivariate $\mathbb{R}^{N_{in}} \to \mathbb{R}^{N_{out}}$ linear regression task, we can consider N_{out} independent modules of the type described before. In this case, the number of trainable parameters is $O(N_{in}^2 \cdot N_{out})$ with $O(N_{in} \cdot N_{out})$ hidden units ($O(N_{in}^2 \cdot N_{out})$ if only tuning the readout weights).

Appendix B. Dynamical adaptation replaces synaptic plasticity: virtual weights can implement rule switching

Consider a task inspired by the Wisconsin card-sorting test. Each card in a deck is defined by two features: color and shape. In each iteration, the agent receives a cue card and two guide cards–one matching the cue by color and the other by shape. The agent must select the correct guide card to receive a positive reward $R(t) = 1$. Initially, the correct choice is based on color matching. However, after 10 trials, the classification rule changes, rewarding shape matching instead. A successful agent must quickly adapt its decision-making to these contextual shifts.

We consider a simple neural network that predicts reward probability for each (cue card, guide card) pair. Two populations, z_1 and z_2, encode similarity based on color and shape, respectively. A third population, $y = \phi(w_1 z_1 + w_2 z_2)$, integrates this information to predict the reward probability. Here $\phi(x) = \text{sigmoid}(x - \text{bias})$ represents the neural transfer function. The weights w_1 and w_2 combine the features to determine the prediction, and the guide card with the highest predicted reward probability is selected. The agent dynamically updates the model based on received rewards to adapt to changing contexts.

Traditionally, weights w_1 and w_2 are considered synaptic weights, and learning is attributed to synaptic plasticity. Alternatively, w_1 and w_2 can be reinterpreted as the activity of two additional populations acting as gain modulators In our model, we refer to them as "virtual weights," as they replace traditional physical synaptic weights, effectively virtualizing the learning process. These modulatory populations dynamically reconfigure the responses of the network, enabling fast, flexible adaptations without the need for synaptic changes. One of the key advantages of this virtualization is that it allows for learning rules that are not constrained by the strict locality requirements inherent to synaptic plasticity.

To support this point, in Fig. A.2 we compare synaptic learning using the gradient-based, delta rule $\Delta \mathbf{w} = \eta e \cdot \mathbf{z}$ with a non-local learning rule that force constraints $w_1(t) + w_2(t) = const$. This results in the update rule $\Delta w_j = \eta(-1)^j e \cdot (z_2 - z_1)$. Here $e(t) = (R(t) - y(t))$ represents the

[4] This can be done assuming that R^{ap}, R_w are diagonal matrices (and thus reducing the problem to estimating N_{in} 1×1 products)

reward prediction error for the last choice and η is the learning rate, controlling the speed of adaptation. As the latter learning rule features non-local terms, it can easily be implemented by a network that dynamically adjusts the activities of two modulatory populations of virtual weights w_1, w_2 without relying on synaptic updates (see Fig. A.2 C). In Fig. A.2 we compare the performance of an agent that learns with synaptic plasticity with an agent that performs dynamic adaptation. As we can see in panel Fig. A.2 F, decreasing the timescales of the learning process, the network with dynamic adaptation achieves maximum reward after seeing only one in-context example, while the performance of the network with synaptic plasticity plateaus at near chance level. This is supported by the psychometric curve shown in Fig. A.2 G. Interestingly, the comparison between the two curves in Fig. A.2 G closely mirrors findings from a recent study (Courellis et al., 2024). In that study, some human participants displayed the ability to adjust their behavior after just one example, while others, with impaired cognitive abilities, had to relearn the previous classification rule through trial and error from scratch.

Appendix C. Additional details on the "dynamical learning of a temporal trajectory" experiment

In the "dynamical learning of a temporal trajectory" experiment we test our architecture and non-synaptic learning approach on temporal tasks. The primary goal is to analyze the network's ability to generalize beyond the target frequencies used to pre-train our networks. Three networks are defined in this experiment: one uses a reservoir to compute temporal features and encode the target temporal trajectory, one predicts the required weight updates, and one predicts the scalar product, as discussed in the main text.

For clarity here we report the entire experiment protocol divided in the two different phases:

- **Phase 1: development of the network structure (algorithm distillation)**
  - We simulate the learning dynamics by integrating Eq. (8) with a suitable discretization timestep dt. We collect the trajectories $\{(\mathbf{z}_\alpha(t), \mathbf{w}_\alpha(t), e_\alpha(t)) | \alpha \in \mathcal{A}_{ID}, t \in \{0, dt, \cdots, T\}\}$.
  - We tune the readout weights Θ_y, Θ_w of the two GMRNN networks Y_{Θ_y}, W_{Θ_w} to minimize, respectively, the objectives

    $$L_Y(\Theta_y) = \sum_{\alpha,t} \left| Y_{\theta_y}(\{\mathbf{z}_\alpha(t), \mathbf{w}_\alpha(t)\}) - \mathbf{w}_\alpha(t) \cdot \mathbf{z}_\alpha(t) \right|^2$$

    and

    $$L_W(\Theta_w) = \sum_{\alpha,t} \left| W_{\Theta_w}(\{\mathbf{z}_\alpha(t), e_\alpha(t)\}) - \frac{d\mathbf{w}_i(t)}{dt} \right|^2$$

    this is done through a pseudo-inverse computation.
- **Phase 2: dynamical learning**
  - **(Open loop)** The readout of the reservoir network is replaced with Y. All the synaptic weights are fixed. Given a novel sequence $y^{targ}_\alpha(t)$. Evolve neural dynamics following Eq. (9). The neural activity $\mathbf{w}_\alpha(t)$ coding for the virtual weights is driven by W_{Θ_w}.
  - **(Closed loop)** The external input is removed, the neural activity $\mathbf{w}_\alpha(t)$ coding for the virtual weights is frozen and $y(t) = Y_{\Theta_y}(z_\alpha(t), w_\alpha(t))$ is fed back as input

Training parameters. We used for training five target trajectories $y^{targ}(t) = 0.8 \sin(\omega_{targ} t)$, with five angular velocities, ω_{targ}, ranging from 0.04 to 0.08. The parameters used in the simulation are summarized in Table C.1. Inputs are projected to the network through Gaussian weights with zero mean and variance σ^2_{in} to distribute the input information across multiple units in the reservoir.

Loss function. The online learning dynamics (Eq. (8)) for the readout weights $\mathbf{w}$ that the networks need to replicate is given by online gradient descent over the instantaneous reconstruction error:

$$L(\mathbf{w}, t) = |y(t) - y^{targ}(t)|^2 \quad \rightarrow \quad \dot{\mathbf{w}}_j = -\eta \frac{\partial L}{\partial \mathbf{w}_j} = -\eta \big(y(t) - y^{\text{target}}(t)\big) \mathbf{z}_j(t),$$

where η is the learning rate.

Table C.1
Simulation parameters.

Param	Symbol	Res	Grad Net	Scalar Net
Network Size	N	20	500	500
Input Dimension	I	1	100 + 10	100+100
Apical Input Dimension	I^{ap}	0	1	100
Output Dimension	O	1	100	1
Time Step	dt	0.005		
Reservoir Time Constant	τ_{m_f}	10 dt	1 dt	1 dt
Input weights var	σ_{input}	0.06	0.06	0.06
Apical Input weights var	σ^{ap}_{input}	0.0	0.1	0.1
Recurrent weights	σ_{rec}	0.99 / $\sqrt{N}$	0.5 / $\sqrt{N}$	0. / $\sqrt{N}$
Gain-modulation factor	γ_{net}	0.	1.	1.

Evaluation. Performances are evaluated by measuring the MSE between the target trajectory ant the predicted one $y(t)$, for different values of the trajectory angular velocity, equally distribute between 0.01 and 0.1.

Appendix D. Additional details on the reinforcement learning experiments

For clarity here we report the entire experiment protocol divided into the two different phases:

- **Phase 1: development of the network structure (algorithm distillation)**
  - For each game $i \in [N_{games}]$ we select a random environment α_i uniformly form the ID set $\alpha_i \sim Unif(\mathcal{A}_{ID})$. We then simulate an agent playing the game with policy $\boldsymbol{\pi} = \text{Softmax}(Y(\mathbf{z}, \mathbf{w}))$, coupled with a policy gradient learning dynamics for the weights $\mathbf{w}$ (Eq. (12) or (D.1) with a timestep dt) until the termination condition is reached for that game at time T_i. We collect the trajectory $\{(\mathbf{z}_i(t), \mathbf{w}_i(t), \boldsymbol{\pi}_i(t), r_i(t), a_i(t)) | i \in [N_{games}], t \in \{0, dt, \cdots, T_i\}\}$.
  - We tune the readout weights Θ_y, Θ_w of the two GMRNN Y_{Θ_y}, W_{Θ_w} to minimize ,respectively, the objectives

    $$L_Y(\Theta_y) = \sum_{\alpha,t} \left| Y_{\theta_y}(\{\mathbf{z}_\alpha(t), \mathbf{w}_\alpha(t)\}) - \mathbf{w}_\alpha(t) \cdot \mathbf{z}_\alpha(t) \right|^2$$

    and

    $$L_W(\Theta_w) = \sum_{i,t} \left| W_{\Theta_w}(\{r_i(t), a_i(t), \boldsymbol{\pi}_i(t), \mathbf{z}_i(t)\}) - \frac{d\mathbf{w}_i(t)}{dt} \right|^2$$

    this is done through a pseudo-inverse computation.
- **Phase 2: dynamical learning**
  - The agent policy is now parametrized by the network Y_{Θ_y}, while its internal learning dynamics is given by the network W_{Θ_w}. We now extract random environments α_i uniformly form the OOD set $\alpha_i \sim Unif(\mathcal{A}_{OOD})$. For each trial, we simulate the environment dynamics coupled with the network learning dynamics in Eq. (13).
  - **(evaluation)** The virtual weights are freezed. The agent parametrized by the network $Y_{\Theta_y}(\cdot, \mathbf{w})$ is now tested on all environments $\mathcal{A}$.

D.1. Reinforcement learning and the discount factor

Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards. The discount factor, usually denoted by

γ_d (where $0 \leq \gamma_d \leq 1$), is crucial in RL as it determines the importance of future rewards. The cumulative reward R_t at time step t is given by:

$$R_t = r_t + \gamma_d r_{t+1} + \gamma_d^2 r_{t+2} + \gamma_d^3 r_{t+3} + \dots$$

In our experiment, the policy gradient update rule in the presence of the discount factor could be approximated as (see Bellec et al., 2020):

$$\begin{cases} \dot{\mathbf{e}} & = -\frac{\mathbf{e}}{\tau_e} + \left(\mathbb{1}_a - \boldsymbol{\pi}\right) \odot \mathbf{x} \\ \mathbf{y} & = \mathbf{w} \cdot \mathbf{x} \\ \tau_w \dot{\mathbf{w}} & = r\mathbf{e} \end{cases} \tag{D.1}$$

where $\mathbf{e}$ is an exponential temporal filtering of the term $\left(\mathbb{1}_a - \boldsymbol{\pi}\right) \odot \mathbf{x}$, with a timescale $\tau_e = -\frac{1}{\log(\gamma_d)}$. The importance of the discount factor lies in its ability to balance immediate and future rewards. A higher γ_d values future rewards more significantly, encouraging the agent to consider the long-term consequences of its actions. Conversely, a lower γ_d makes the agent prioritize immediate rewards. This balance is essential for the stability of the learning process. An appropriately chosen γ_d ensures stable learning and convergence; if γ_d is too high, the agent might overvalue distant future rewards, leading to instability, while a very low γ_d can result in short-sighted behavior. In the limit $\gamma_d \to 0$ we recover Eq. (12) used in the main text. If the discount factor is zero, the agent's policy would change only when it is very close to the moment the reward is received. In a reaching task, the agent will only change its policy when it is very close to the reward, completely ignoring the need for long-term planning and making it unlikely to ever reach the goal from distant starting points.

To demonstrate that our approach performs the temporal computation required to consider future rewards, we compare the performances of our network (see Fig. A.3A-B) to a network without recurrent connections, which therefore cannot perform temporal computations. Recurrent connections enable a network to maintain a memory of past states and actions, effectively allowing it to use information from previous time steps to inform current decisions. Without these connections, a network operates purely on the current input without any contextual information from prior steps, thus lacking the ability to perform temporal computations (see Fig. A.3D-E).

Performances are higher in the first case. This can be visualized by looking at the policy at the end of the dynamic reinforcement learning for a specific target location. When the recurrent weights are set to zero, the policy points towards the target position only when nearby the target itself (see Fig. A.3F), resulting in failure when the agent randomly moves in the wrong direction at the beginning of the task (see Fig. A.3F, black line).

On the other hand, in the presence of recurrent weights, the proper policy (pointing towards the target, see Fig. A.3C) is known even when far from the target, allowing optimal long-term planning (see Fig. A.3C, black line).

D.2. Additional details on the bandit experiment

Family of tasks. We consider a family of tasks $\mathcal{A} = \{ID, OOD\}$, such that every element $\alpha \in \mathcal{A}$ represents a Bernoulli K-armed bandit problem with rewards $\{\hat{r}_i^\alpha\}_{i=1}^K$. Each task α is specified by a set $P_\alpha \subset \{1 \cdot, K\}$ of positive arms, such that

$$Prob_\alpha(r|i) = \hat{r}_i^\alpha = \begin{cases} p & i \in P_\alpha \\ 1-p & i \notin P_\alpha \end{cases} \tag{D.2}$$

In the main text experiment, we used $K = 10$ and $P_{ID} := \{i \in [K] | i \equiv 0 (\mathrm{mod} 2)\}$ and $P_{OOD} := \{i \in [K] | i \equiv 1 (\mathrm{mod} 2)\}$.

Network details and hyperparameter search. We compare an architecture with gain modulation ($\gamma = 1$) with an architecture without gain modulation $\gamma = 0$. For each of the two settings, we select the best hyperparameters by fixing the number of hidden units ($N_h = 100$) and varying the standard deviation of the projection matrices $R_{ij} \sim \mathcal{N}(0, \sigma_R^2/20)$ $R_{ij}^{ap} \sim \mathcal{N}(0, \sigma_R^2)$, $\log_{10}(\sigma_R) \in \{-2, -1.8, \cdots, 0\}$, the standard deviation of the bias $b_i \sim \mathcal{N}(0, \sigma_b^2)$, $\sigma_b \in \{0, 0.1, \cdots, 1\}$ and the nonlinearity type $\phi \in \{\tanh, \mathrm{softplus}\}$. For each point in the grid, we train 20 models using The best hyperparameters found were used in the subsequent experiments In Fig. 2 B, D the reported regrets are on models trained with $N_h = 100$ hidden features

Regret per round. Given an agent that plays in an environment α receiving rewards $\{r^\alpha(t)\}_{t \in \mathbb{N}_+}$, the regret per round $\rho(T)$ achieved by the agent at round T is defined as:

$$\rho(T) = \mu_\star^\alpha - \frac{1}{T} \sum_{t=1}^{T} r^\alpha(t) \tag{D.3}$$

Where $\mu_*^\alpha := \max\{\hat{r}_k^\alpha | k \in [K]\} = \max\{p, 1-p\}$ is the maximum expected reward that can be obtained in each round playing the optimal policy that selects one of the best arms with probability 1.

D.3. Additional details on the darkroom experiment

This experiment investigates the capability of out dynamical reinforcement learning approach, to improve an agent's ability to navigate a 2D environment. The agent is trained to locate a randomly placed, not observable food source, with its policy evolving over multiple episodes through.

The environment is a 2D grid where an agent starts at the center, aiming to reach a randomly positioned food item. The food positions vary every 600 episodes, introducing. The agent's movements are restricted within the grid, with actions limited to moving left, right, up, or down.

To represent the agent's position, a place cell encoder converts the agent's (x, y) coordinates into a higher-dimensional feature vector using Gaussian functions. This encoding helps in effectively capturing spatial information.

The agent's policy is linear, with action probabilities derived from a softmax function applied to the encoded state. For this reason, only two networks are used for the task, the gradient and the scalar-product network see Table D.2 for details on parameters.

Table D.2
Simulation parameters.

Parameter	Symbol	Gradient Net	Scalar Net
Network Size	N	500	1000
Input Dimension	I	$25+4+10$	$4 \times 25 + 4 \times 25$
Apical Input Dimension	I^{ap}	1	4×25
Output Dimension	O	4×25	4
Time Step	dt	0.005	
Reservoir Time Constant	τ_{m_f}	$1\ dt$	$1\ dt$
Input weights var	σ_{input}	0.01	0.01
Apical Input weights var	σ_{input}	0.1	0.1
Recurrent weights	σ_{rec}	$0.5\ /\ \sqrt{N}$	$0.\ /\ \sqrt{N}$
Gain-modulation factor	γ_{net}	1.	1.

Data collection and pre-training. The policy is updated using the policy gradient method, adjusting weights based on received rewards to improve decision-making over time.

Learning data are collected for 8 different positions of the food, equally distributed on a circle. Two types of networks are used: one to model the agent's state dynamics and another to handle gradient updates necessary for learning. These networks influence the agent's internal state and learning process, enabling policy refinement. The two networks readout are trained with a linear regression on their readout weight, to reproduce the proper weight updates an scalar products.

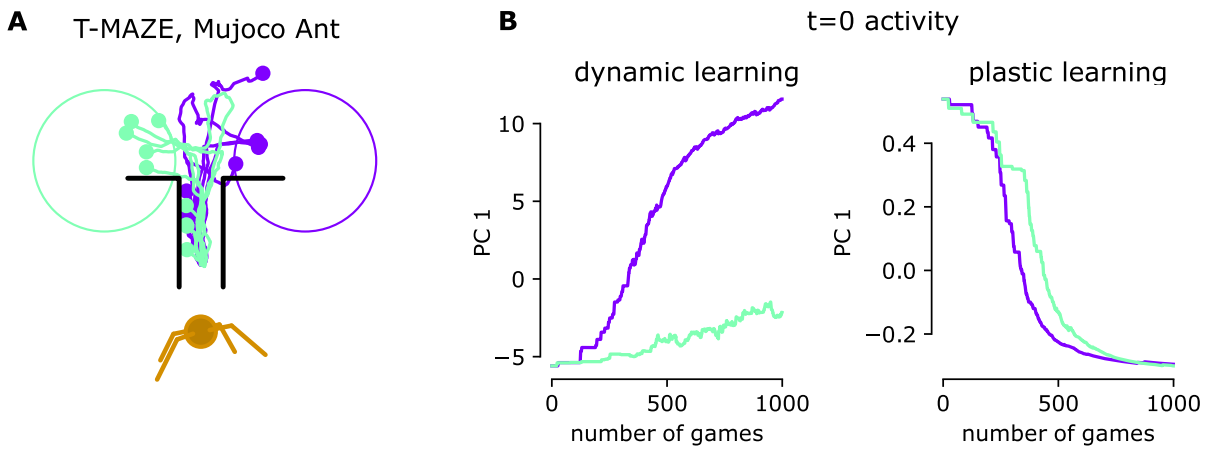

Fig. E.4. T-Maze task in Mujoco-ANT environment. A. Task description, the ant has to reach a target location after an horizontal corridor. **B.** Decoding of contextual information from the network activity at the initial time of each trial/game, as a function of the trial/game number in the case of dynamic and plastic learning (left and right respectively).

Dynamical reinforcement learning and evaluation. The agent is tested again on the 8 positions used for pre-training, and on 8 new positions. Similarly to previous experiments, policy gradient is no longer used and replaced by virtual weights updated estimated by the gradient network, and used by the scalar-product network to predict agent policy.

Performance is measured by the total reward accumulated over episodes. Visualizing the agent's trajectories reveals its navigation efficiency and decision-making process.

Appendix E. Estimating the occurrence of dynamic adaptation

To evaluate the ability of our biologically plausible recurrent network to adapt without synaptic plasticity, we tested it in a context-dependent locomotion task within the MuJoCo simulation environment. The task requires an Ant agent to reach a target location positioned either to the left or right at the end of a horizontal corridor. At the beginning of each trial, the ant does not have prior explicit information about the target direction and must infer it dynamically as the task unfolds.

E.1. Decoding contextual information from network activity

To understand how contextual information is processed, we analyzed the network's activity at the initial time step of each trial. We compared two learning paradigms: Dynamic adaptation (left panel, Fig. E.4B): In this case, the contextual information–whether the food is to the left or right–is encoded in the neuronal activity, rather than in fixed synaptic weights. This allows the network to flexibly adjust its response from trial to trial without requiring long-term structural changes. Plastic adaptation (right panel, Fig. E.4B): Here, learning occurs through synaptic weight updates. As a result, at the beginning of each trial, contextual information is encoded in the synaptic weights and not directly in the network activity.

Implications for neural encoding and experimental validation. At the start of each trial, the ant must initially move straight down the corridor before making a directional choice. Given this, it is not immediately obvious that any information about the correct target location should be present in the network's state at the very beginning of the trial. Indeed, if contextual information were stored only in synaptic weights, it would not be explicitly represented in early network activity (as shown in Fig. B, right panel). However, when information about the subtask is dynamically encoded in neuronal activity, it becomes detectable through neural recordings (Fig. B, left panel). This finding suggests that adaptive behavior in biological systems may rely more on network dynamics and transient activity states, rather than purely on synaptic modifications. Our results highlight a fundamental distinction between synaptic plasticity-based adaptation and dynamic adaptation via network activity, with important implications for understanding biological learning mechanisms and designing adaptive artificial systems.

References

Abraham, W. C., Jones, O. D., & Glanzman, D. L. (2019). Is plasticity of synapses the mechanism of long-term memory storage? *npj Science of Learning*, *4*(1), 9.

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S. et al. (2023). Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Ba, J., Hinton, G. E., Mnih, V., Leibo, J. Z., & Ionescu, C. (2016). Using fast weights to attend to the recent past. *Advances in Neural Information Processing Systems*, *29*.

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., & Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, *11*(1), 1–15.

Berry, D. A., & Fristedt, B. (1985). Bandit problems: Sequential allocation of experiments (monographs on statistics and applied probability). London: Chapman and Hall, 5(71–87), 7.

Braun, U., Schäfer, A., Walter, H., Erk, S., Romanczuk-Seiferth, N., Haddad, L., Schweiger, J. I., Grimm, O., Heinz, A., Tost, H. et al. (2015). Dynamic reconfiguration of frontal brain networks during executive cognition in humans. *Proceedings of the National Academy of Sciences*, *112*(37), 11678–11683.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, *33*, 1877–1901.

Capone, C., Lupo, C., Muratore, P., & Paolucci, P. S. (2023). Beyond spiking networks: The computational advantages of dendritic amplification and input segregation. *Proceedings of the National Academy of Sciences*, *120*(49), e2220743120.

Capone, C., & Paolucci, P. S. (2022). Towards biologically plausible dreaming and planning. arXiv preprint arXiv:2205.10044.

Cardin, J. A., Palmer, L. A., & Contreras, D. (2008). Cellular mechanisms underlying stimulus-dependent gain modulation in primary visual cortex neurons in vivo. *Neuron*, *59*(1), 150–160.

Chalvidal, M., Serre, T., & VanRullen, R. (2022). Meta-reinforcement learning with self-modifying networks. *Advances in Neural Information Processing Systems*, *35*, 7838–7851.

Courellis, H. S., Minxha, J., Cardenas, A. R., Kimmel, D. L., Reed, C. M., Valiante, T. A., Salzman, C. D., Mamelak, A. N., Fusi, S., & Rutishauser, U. (2024). Abstract representations emerge in human hippocampal neurons during inference. *Nature*, (pp. 1–9).

Daw, N. D., Niv, Y., & Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, *8*(12), 1704–1711.

De Pittà, M., & Brunel, N. (2022). Multiple forms of working memory emerge from synapse–astrocyte interactions in a neuron–glia network model. *Proceedings of the National Academy of Sciences*, *119*(43), e2207912119.

Ferguson, K. A., & Cardin, J. A. (2020). Mechanisms underlying gain modulation in the cortex. *Nature Reviews Neuroscience*, *21*(2), 80–92.

Feulner, B., Perich, M. G., Miller, L. E., Clopath, C., & Gallego, J. A. (2022). Feedback-based motor control can guide plasticity and drive rapid learning. *bioRxiv*, 2022–10.

Gu, A., & Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752.

Gu, A., Goel, K., & Re, C. (2021). Efficiently modeling long sequences with structured state spaces. In *International conference on learning representations*.

Guarino, D., Carannante, I., & Destexhe, A. (2025). Neuromodulators control neuronal dynamics through feature space reshaping. *bioRxiv*, 2025–05.

Guerguiev, J., Lillicrap, T. P., & Richards, B. A. (2017). Towards deep learning with segregated dendrites. *eLife*, *6*, e22901.

Hart, A., Hook, J., & Dawes, J. (2020). Embedding and approximation theorems for echo state networks. *Neural Networks*, *128*, 234–247.

Heald, J. B., Lengyel, M., & Wolpert, D. M. (2021). Contextual inference underlies the learning of sensorimotor repertoires. *Nature*, *600*(7889), 489–493.

Heald, J. B., Lengyel, M., & Wolpert, D. M. (2023a). Contextual inference in learning and memory. *Trends in cognitive sciences*, *27*(1), 43–64.

Heald, J. B., Wolpert, D. M., & Lengyel, M. (2023b). The computational and neural bases of context-dependent learning. *Annual Review of Neuroscience*, *46*, 233–258. https://doi.org/10.1146/annurev-neuro-092322-100402

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*, *70*(1–3), 489–501.

Irie, K., Schlag, I., Csordás, R., & Schmidhuber, J. (2021). Going beyond linear transformers with recurrent fast weight programmers. *Advances in Neural Information Processing Systems*, *34*, 7703–7717.

Jiang, L. P., & Rao, R. P. N. (2024). Dynamic predictive coding: A model of hierarchical sequence learning and prediction in the neocortex. *PLoS Computational Biology*, *20*(2), e1011801.

Köksal-Ersöz, E., Chossat, P., & Lavigne, F. (2024). Gain modulation of actions selection without synaptic relearning. arXiv preprint arXiv:2501.01964.

Larkum, M. (2013a). A cellular mechanism for cortical associations: An organizing principle for the cerebral cortex. *Trends in Neurosciences*, *36*(3), 141–151.

Larkum, M. (2013b). A cellular mechanism for cortical associations: An organizing principle for the cerebral cortex. *Trends in Neurosciences*, *36*(3), 141–151. https://doi.org/10.1016/j.tins.2012.11.006

Laskin, M., Wang, L., Oh, J., Parisotto, E., Spencer, S., Steigerwald, R., Strouse, D. J., Hansen, S., Filos, A., Brooks, E. et al. (2022). In-context reinforcement learning with algorithm distillation. arXiv preprint arXiv:2210.14215.

Lee, I., & Lee, C.-H. (2013). Contextual behavior and neural circuits. *Frontiers in Neural Circuits*, *7*, 84.

McDougle, S. D., Bond, K. M., & Taylor, J. A. (2015). Explicit and implicit processes constitute the fast and slow processes of sensorimotor learning. *Journal of Neuroscience*, *35*(26), 9568–9579.

Meulemans, A., Zucchet, N., Kobayashi, S., Von Oswald, J., & Sacramento, J. (2022). The least-control principle for local learning at equilibrium. *Advances in Neural Information Processing Systems*, *35*, 33603–33617.

Miconi, T., Stanley, K., & Clune, J. (2018). Differentiable plasticity: Training plastic neural networks with backpropagation. In *International conference on machine learning* (pp. 3559–3568). PMLR.

Millidge, B., Tang, M., Osanlouy, M., Harper, N. S., & Bogacz, R. (2024). Predictive coding networks for temporal prediction. *PLoS Computational Biology*, *20*(4), e1011183.

Morris, R. G. M. (1981). Spatial localization does not require the presence of local cues. *Learning and Motivation*, *12*(2), 239–260.

O'Doherty, J. P., Cockburn, J., & Pauli, W. M. (2017). Learning, reward, and decision making. *Annual Review of Psychology*, *68*(1), 73–100.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A. et al. (2022). In-context learning and induction heads. arXiv preprint arXiv:2209.11895.

Orvieto, A., Smith, S. L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., & De, S. (2023). Resurrecting recurrent neural networks for long sequences. In *International conference on machine learning* (pp. 26670–26698). PMLR.

Payeur, A., Guerguiev, J., Zenke, F., Richards, B. A., & Naud, R. (2021). Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature Neuroscience*, *24*(7), 1010–1019.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., & Courville, A. (2018). Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*. (*32*).

Poirazi, P., & Papoutsi, A. (2020). Illuminating dendritic function with computational models. *Nature Reviews Neuroscience*, *21*(6), 303–321.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). Zero-shot text-to-image generation. In *International conference on machine learning* (pp. 8821–8831). PMLR.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G. K. et al. (2020). Hopfield networks is all you need. arXiv preprint arXiv:2008.02217.

Romera-Paredes, B., & Torr, P. (2015). An embarrassingly simple approach to zero-shot learning. In *International conference on machine learning* (pp. 2152–2161). PMLR.

Sacramento, J., Ponte Costa, R., Bengio, Y., & Walter, S. (2018). Dendritic cortical microcircuits approximate the backpropagation algorithm. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems 31* (pp. 8721–8732). Curran Associates, Inc.

Schmidhuber, J. (1992). Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, *4*(1), 131–139.

Shai, A. S., Anastassiou, C. A., Larkum, M. E., & Koch, C. (2015). Physiology of layer 5 pyramidal neurons in mouse primary visual cortex: Coincidence detection through bursting. *PLoS Computational Biology*, *11*(3), e1004090.

Singh, A., Chan, S., Moskovitz, T., Grant, E., Saxe, A., & Hill, F. (2024). The transient nature of emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, *36*.

Standage, D. I., Areshenkoff, C. N., Gale, D. J., Nashed, J. Y., Flanagan, J. R., & Gallivan, J. P. (2023). Whole-brain dynamics of human sensorimotor adaptation. *Cerebral Cortex*, *33*(8), 4761–4778.

Stroud, J. P., Porter, M. A., Hennequin, G., & Vogels, T. P. (2018). Motor primitives in space and time via targeted gain modulation in cortical networks. *Nature Neuroscience*, *21*(12), 1774–1783.

Urbanczik, R., & Walter, S. (2014). Learning by the dendritic prediction of somatic spiking. *Neuron*, *81*(3), 521–528.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D. et al. (2016). Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, *29*.

Von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., & Vladymyrov, M. (2023). Transformers learn in-context by gradient descent. In *International conference on machine learning* (pp. 35151–35174). PMLR.

Wainstein, G., Whyte, C. J., Martens, K. A. E., Müller, E. J., Medel, V., Anderson, B., Stöttinger, E., Danckert, J., Munn, B. R., & Shine, J. M. (2025). Evidence from pupillometry, fMRI, and RNN modelling shows that gain neuromodulation mediates task-relevant perceptual switches. *eLife*, *13*, RP93191.

Walter, S., Dold, D., Kungl, A. F., Ellenberger, B., Jordan, J., Bengio, Y., Sacramento, J., & Petrovici, M. A. (2024). A neuronal least-action principle for real-time learning in cortical circuits. *eLife*, *12*, RP89674.

Wilson, H. R., & Cowan, J. D. (1972). Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical Journal*, *12*(1), 1–24.

Wybo, W. A. M., Tsai, M. C., Tran, V. A. K., Illing, B., Jordan, J., Morrison, A., & Walter, S. (2023). Nmda-driven dendritic modulation enables multitask representation learning in hierarchical sensory processing pathways. *Proceedings of the National Academy of Sciences*, *120*(32), e2300558120.

Yildiz, I. B., Jaeger, H., & Kiebel, S. J. (2012). Re-visiting the echo state property. *Neural Networks*, *35*, 1–9.

Zhang, R., Frei, S., & Bartlett, P. L. (2024). Trained transformers learn linear models in-context. *Journal of Machine Learning Research*, *25*(49), 1–55.

Zucchet, N., Kobayashi, S., Akram, Y., Von Oswald, J., Larcher, M., Steger, A., & Sacramento, J. (2023). Gated recurrent neural networks discover attention. arXiv preprint arXiv:2309.01775.