Differentiable Plastic Recurrent Neural Networks: A Systematic Evaluation

Anonymous authors
Paper under double-blind review

Abstract

Plastic recurrent neural networks (RNNs) augment standard architectures with weights that adapt during inference, offering a biologically-inspired path toward more adaptive artificial systems. Despite a growing collection of architectures, the field lacks a systematic account of how these methods relate to one another, when plasticity improves performance, and whether learned plastic components are actually used at inference time. We address this gap by placing existing differentiable plastic RNNs under a unified notation, conducting a parameter matched evaluation of ten architectures across four task families, and analyzing how these architectures utilize plasticity, if at all. We also identify an underexplored design direction for plastic RNNs. Grounding plasticity in extended neuron dynamics and population-level synchronization, rather than instantaneous pre- and post-synaptic correlations, we propose the Plastic Continuous Thought Machine (Plastic CTM). On recall and continual learning tasks, plastic and non-plastic architectures perform comparably, but on in-context learning, plastic architectures consistently outperform their non-plastic counterparts, with the Plastic CTM achieving the strongest results among baselines. Our analysis reveals that architectures with similar accuracy can use plasticity in very different ways, with some relying on plastic weights for genuine in-context adaptation, whilst others achieve comparable accuracy even with their plasticity disabled. These findings highlight the need for more discriminating benchmarks, mechanistic analysis, and exploration of richer biologically inspired rules such as population level synchronization to meaningfully advance the field.

1 Introduction

Organisms in nature exhibit remarkable adaptability, enabling them to learn from experience and adjust their behavior to changing environments over extended timescales. This adaptability is supported by synaptic plasticity, the primary basis for long-term learning and memory in the brain, where connection strengths adjust as a function of ongoing neural activity (Martin et al., 2000). Achieving comparable adaptivity is a central challenge in machine learning. One popular approach is meta-learning, where models are trained across many tasks so they can rapidly adapt to a new task from limited experience. A prominent family achieves this through gradient-based optimization, adapting via explicit gradient steps at test time (Finn et al., 2017). This approach, however, requires complex training procedures and supervision signals during inference. An alternative method is to adapt through model dynamics alone, with a recurrent neural network (RNN) learning a learning algorithm that can be applied without supervision.

Plastic RNNs, which fall into this second category, offer a path inspired directly by biological synaptic plasticity. These methods contain weights that adapt during inference based on the model's own activity, without requiring gradient computation or explicit loss signals at test time. A major milestone in this direction was the introduction of differentiable plasticity (Miconi, 2016; Miconi et al., 2018), a mechanism that enables RNNs to learn a Hebbian update rule end-to-end via backpropagation. Follow-up work (Miconi et al., 2020) extended this idea with neuromodulation, allowing the network to learn when to enable plasticity, analogous to neuromodulators such as dopamine in the brain. More recently, Short-Term Plasticity Neurons (STPN; Rodriguez et al., 2022) enabled models to learn per-synapse decay rates, effectively learning to

forget. Other lines of work have approached rapid weight adaptation through associative memory (Ba et al., 2016a), hypernetworks (Ha et al., 2016), and gradient-based internal updates (Duan et al., 2023).

While the field has produced a diverse set of architectures and plasticity rules, much less attention has been paid to how these models actually leverage their plasticity at inference time. Performance comparisons alone cannot distinguish between a method that uses plasticity for genuine in-context adaptation and one that effectively learns to disable it. As the field grows, this gap in mechanistic analysis makes it increasingly difficult to identify which design choices matter and where progress is genuinely being made.

In this work, we conduct an extensive evaluation of plastic RNNs. We compare ten plastic and non-plastic architectures across algorithmic, in-context learning, continual learning, and meta-reinforcement learning (meta-RL) tasks, using systematic hyperparameter tuning and rigorous parameter-matching across all baselines. Among the evaluated architectures, we introduce the Plastic Continuous Thought Machine (Plastic CTM), a new plastic RNN inspired by biological evidence that synaptic plasticity is shaped by extended firing histories and population-level synchronization rather than instantaneous correlations alone (Wespatat et al., 2004; Jutras & Buffalo, 2010; Einarsson et al., 2017). Building on the Continuous Thought Machine (Darlow et al., 2025), a recently proposed bio-inspired RNN whose representations naturally capture these properties, the Plastic CTM grounds plasticity in its neuron dynamics and synchronization.

Our evaluation yields several findings about the state of the field. Across algorithmic recall and continual learning tasks, plastic and non-plastic architectures perform comparably. On in-context learning tasks, by contrast, plastic architectures consistently outperform their non-plastic counterparts, with the Plastic CTM achieving the strongest results among evaluated baselines, matching or exceeding meta-learning baselines such as MAML (Finn et al., 2017). On meta-RL, we find that the canonical maze benchmark from Miconi et al. (2018) is saturated under modern training algorithms, with all but one of the evaluated architectures achieving near-identical performance. This suggests that the field requires more discriminating benchmarks to meaningfully evaluate plastic RNNs going forward. Beyond benchmark performance, mechanistic analysis reveals that architectures with similar accuracy can use plasticity in fundamentally different ways. Some methods rely on their plastic weights for genuine in-context adaptation, while others achieve comparable accuracy without using their plasticity at all, and even among those that do rely on it, the underlying weight and hidden-state dynamics can differ substantially. Our findings point to grounding plasticity in neuron dynamics as a promising direction, and to mechanistic analysis as an essential complement to benchmark accuracy when evaluating plastic RNNs.

In summary, our contributions are as follows:

- We describe existing differentiable plastic RNNs under a unified notation and conduct a systematic, parameter-matched evaluation of ten architectures across algorithmic recall, in-context learning, continual learning, and meta-RL tasks.
- We identify grounding plasticity in extended neuron dynamics and population-level synchronization as an underexplored design direction, and instantiate it with the Plastic Continuous Thought Machine (Plastic CTM).
- Through mechanistic analysis, we show that architectures with similar benchmark performance can use plasticity in fundamentally different ways.

2 Background

2.1 Synaptic Plasticity

Synaptic plasticity is thought to be the primary basis for long-term learning and memory in the brain (Martin et al., 2000). Hebb's rule, an early theory of this mechanism, proposes that connections strengthen when one neuron repeatedly contributes to firing another, often summarized as "neurons that fire together, wire together" (Hebb, 2005). Other notable Hebbian learning rules, where synaptic updates are a function of pre- and post-synaptic activity, include Oja's rule (Oja, 1982) for improved stability, and Spike-Timing

Dependent Plasticity (STDP), incorporating the relative timing of spikes to modulate synaptic changes (Bi & Poo, 1998).

However, these purely pre-post Hebbian rules oversimplify biological plasticity. For instance, plasticity in the brain is actively gated by neuromodulatory systems, resulting in "three-factor" (modulated) Hebbian rules, with chemicals such as dopamine determining when and where synaptic changes are consolidated (Calabresi et al., 2007; Yagishita et al., 2014). This allows the brain to filter out irrelevant events while selectively committing important information to memory, and is thought to play a central role in reward-based learning (Schultz et al., 1997). Another example is the influence of population-level dynamics on plasticity. Gamma-band oscillations operate at the same timescale as STDP, and the synchronization between populations of neurons modulates its effect (Wespatat et al., 2004; Jutras & Buffalo, 2010). More recent work has proposed modified STDP rules that incorporate both synchronization and extended temporal firing histories (Einarsson et al., 2017). Neuromodulation has been incorporated into differentiable plastic RNNs through the backpropamine framework (Miconi et al., 2020), reviewed in Section 3.2. Synchronization-driven plasticity has been explored in spiking neural networks (Tian et al., 2025; Bej et al., 2025; Chandrasekharan et al., 2026) but not in differentiable plastic RNNs. We address this gap in Section 4.

2.2 Plastic Neural Networks

Inspired by synaptic plasticity in biological brains, plastic deep neural networks modify their weights based on ongoing neural activity to enable rapid adaptation and learning. An early instantiation of this idea is *fast weights* (FW; Hinton & Plaut, 1987; Schmidhuber, 1992), where a slow-updating network generates changes in the weights of a fast-changing network. Schmidhuber (1992) proposed a Hebbian-style[1] variant in which the fast weight updates take the form of an outer product of two activation patterns produced by the slow network. These early methods foreshadow much of the subsequent work on plastic neural networks, with later methods broadly falling into hypernetwork-style approaches that generate weights directly (Ha et al., 2016) and Hebbian-style approaches that update weights via outer products of activations (Ba et al., 2016a; Miconi et al., 2018). One line of work uses evolutionary algorithms to discover learning mechanisms, with evolved plastic networks demonstrating strong capabilities in non-stationary reward-based tasks (Soltoggio et al., 2018) and robustness to noise (Najarro & Risi, 2020; Leung et al., 2025). A complementary line trains plastic networks end-to-end by gradient descent, with the dominant approach following the differentiable Hebbian plasticity framework of Miconi et al. (2018), extended with neuromodulated plasticity (Miconi et al., 2020) and short-term plasticity dynamics (Rodriguez et al., 2022). We review these differentiable plastic RNNs in detail in Section 3.

Beyond this differentiable plasticity framework, modern implementations of FW store recent hidden states in a decaying associative memory matrix and attend to the recent past through inner-loop refinement of the hidden state (Ba et al., 2016a; Munkhdalai & Trischler, 2018). Hyper Long Short-Term Memory (HyperLSTM; Ha et al., 2016) networks use a "hypernetwork" to generate the main network's weights conditioned on the current hidden state. Gradient-based plasticity (Duan et al., 2023) computes weight updates by differentiating a learned internal loss at inference time. Finally, other works have explored making the synapses themselves recurrent neural networks (Kirsch & Schmidhuber, 2021; Pedersen et al., 2024).

2.3 Meta-Learning

Meta-learning, or "learning to learn", aims to build agents that rapidly adapt from ongoing experience (Schmidhuber et al., 1997; Bengio et al., 2013), with approaches often categorized into three families (Hospedales et al., 2021). Model-based methods adapt through internal or external memory, for example, LSTMs (Hochreiter & Schmidhuber, 1997; Gers et al., 2000) solving meta-RL tasks entirely through hidden state changes (Wang et al., 2016; Duan et al., 2016). Metric-based methods like ProtoNets (Snell et al., 2017) classify by learned similarity to support samples, and optimization-based methods like MAML (Finn et al., 2017) learn parameter initializations enabling fast adaptation via gradient descent. Plastic neural networks

[1] We use "Hebbian-style" to denote weight update rules that use an outer product of local activations, e.g., autoassociative updates, rather than strictly pre- and post-activations.

extend the model-based approach by enabling weight adaptation through learned local rules, combining the computational efficiency of internal adaptation with the representational flexibility of modifiable connections.

3 Differentiable Hebbian Plasticity

In this section we review the differentiable Hebbian plastic RNNs, namely the Plastic RNN (Miconi et al., 2018), the Plastic LSTM (Miconi et al., 2020), and STPN (Rodriguez et al., 2022). Each augments a recurrent weight matrix W with a plastic component P^t that evolves online, giving an effective weight at time t of

$$G^t = W + P^t, \tag{1}$$

where methods differ in how P^t is updated. For clarity we write rules with $W \in \mathbb{R}^{d\times d}$ and a generic activation $x^t \in \mathbb{R}^d$, but the updates are architecture-agnostic and apply to other weight matrices by adjusting shapes accordingly. For example, x^t may be the hidden state in some works, or a concatenation of the input and hidden state in others. A summary appears in Table 1.

3.1 Differentiable Plasticity and Plastic RNNs

Leveraging gradient descent and backpropagation, Miconi et al. (2018) introduced "differentiable plasticity", the first method to make Hebbian-style updates trainable end-to-end. The plastic component P^t is updated as an exponential moving average of the outer product of pre and post-synaptic activity, scaled element-wise by a learned per-synapse plasticity coefficient,

$$P^{t+1} = \eta\, \alpha \odot (x^{t-1} \otimes x^t) + (1-\eta)\, P^t, \tag{2}$$

where $\alpha \in \mathbb{R}^{d\times d}$ is the per-synapse plasticity coefficient matrix, $\eta \in \mathbb{R}$ is a learned scalar plasticity learning rate, $\odot$ denotes element-wise multiplication, and $\otimes$ denotes the outer product. The fixed weights W, plasticity coefficients α, and learning rate η are all trained end-to-end by gradient descent.

The framework also accommodates alternative biologically-inspired update rules. For example, the Oja's rule variant (Oja, 1982) replaces the exponential decay with a self-normalizing term that prevents unbounded weight growth,

$$P^{t+1} = P^t + \eta \left[\alpha \odot (x^{t-1} \otimes x^t) - \big(\mathbf{1} \otimes (x^t \odot x^t)\big) \odot P^t\right], \tag{3}$$

where $\mathbf{1} \in \mathbb{R}^d$ is the all-ones vector. By choosing different update rules, the differentiable plasticity framework can implement a wide range of synaptic learning mechanisms while remaining end-to-end trainable.

3.2 Backpropamine and Plastic LSTMs

A limitation of the Plastic RNN update is that plasticity occurs uniformly across time, with every pre/post-synaptic correlation contributing at a rate fixed by the learned scalar η. Miconi et al. (2020) address this with *backpropamine*, which replaces η with a network-computed, time-varying neuromodulatory signal $M^t \in \mathbb{R}^d$, allowing the network to decide when plasticity should occur, inspired by dopaminergic systems that gate plasticity by context (Calabresi et al., 2007). They also replace the soft decay used in Miconi et al. (2018) with a hard clip on the plastic trace, which preserves Hebbian traces between updates rather than continuously eroding them. Writing the plastic component as $P^t = \alpha \odot \tilde{P}^t$ with $\tilde{P}^t \in \mathbb{R}^{d\times d}$, the "three-factor" backpropamine update becomes

$$\tilde{P}^{t+1} = \mathrm{Clip}\big(\tilde{P}^t + (\mathbf{1} \otimes M^t) \odot (x^{t-1} \otimes x^t)\big), \tag{4}$$

with M^t computed from the hidden state and $\mathrm{Clip}(\cdot)$ bounding the trace $\in [-1, 1]$.

Miconi et al. (2020) proposed two instantiations of this scheme. In *simple neuromodulation*, M^t is a learned projection of the current hidden state, gating plasticity on a per-step basis. In *retroactive neuromodulation*, the instantaneous outer product is replaced by a decaying eligibility trace $E^t \in \mathbb{R}^{d\times d}$ that accumulates recent

Table 1: Plastic update rules across methods. All compute an effective weight $G^t = W + P^t$ and differ primarily in how P^t is updated. The Plastic LSTM clips the unscaled trace $\tilde{P}^t$ (with $P^t = \alpha \odot \tilde{P}^t$), and the Plastic CTM additionally normalizes G^t by its Frobenius norm.

Method	Plastic update
Hebbian	$P^{t+1} = \eta\, \alpha \odot (x^{t-1} \otimes x^t) + (1-\eta)\, P^t$
Oja	$P^{t+1} = P^t + \eta \left[\alpha \odot (x^{t-1} \otimes x^t) - (\mathbf{1} \otimes (x^t \odot x^t)) \odot P^t\right]$
Simple Mod.	$\tilde{P}^{t+1} = \text{Clip}\big(\tilde{P}^t + (\mathbf{1} \otimes M^t) \odot (x^{t-1} \otimes x^t)\big)$
Retroactive Mod.	$\tilde{P}^{t+1} = \text{Clip}\big(\tilde{P}^t + (\mathbf{1} \otimes M^t) \odot E^t\big)\,, \quad E^{t+1} = (1-\eta)E^t + \eta(x^{t-1} \otimes x^t)$
STPN	$P^{t+1} = \Gamma \odot (x^{t-1} \otimes x^t) + (\mathbf{1} \otimes \mathbf{1} - \Lambda) \odot P^t$
Plastic CTM (Ours)	$P^{t+1} = \frac{\beta \lVert W \rVert_F}{\lVert S^{t+1} \rVert_F} S^{t+1}, \;\; S_{ij}^{t+1} \propto \sum_{\tau=1}^{t+1} e^{-r_{ij}(t+1-\tau)} z_i^\tau z_j^\tau$

pre/post-synaptic correlations. The neuromodulator gates the conversion of this trace into a weight update,

$$E^{t+1} = (1-\eta)\, E^t + \eta\, (x^{t-1} \otimes x^t), \tag{5}$$

$$\tilde{P}^{t+1} = \text{Clip}\big(\tilde{P}^t + (\mathbf{1} \otimes M^t) \odot E^t\big)\,. \tag{6}$$

This variant is inspired by evidence that dopamine retroactively gates recent Hebbian activity within a short time window (Yagishita et al., 2014), and gives the network access to extended temporal information when deciding whether to commit a weight change. Miconi et al. (2020) also apply the framework to LSTMs, making only the recurrent weights contributing to the candidate cell update plastic (with x^{t-1} the previous hidden state and x^t the candidate cell update) while leaving the forget, input, and output gates non-plastic.

3.3 Short-Term Plasticity Neurons

Short-Term Plasticity Neurons (STPN; Rodriguez et al., 2022) extend the differentiable Hebbian framework with per-synapse decay rates alongside per-synapse learning rates. The plastic component evolves as

$$P^{t+1} = \Gamma \odot (x^{t-1} \otimes x^t) + (\mathbf{1} \otimes \mathbf{1} - \Lambda) \odot P^t, \tag{7}$$

where $\Gamma \in \mathbb{R}^{d \times d}$ is a per-synapse Hebbian learning rate matrix and $\Lambda \in \mathbb{R}^{d \times d}$ is a per-synapse decay rate matrix. Unlike the framework of Miconi et al. (2018), the scalar learning rate η becomes the per-synapse matrix Γ and the scalar decay $(1-\eta)$ becomes the per-synapse matrix $\mathbf{1} \otimes \mathbf{1} - \Lambda$, enabling each connection to learn its own Hebbian learning rate and forgetting timescale. This per-synapse control over learning and forgetting timescales gives STPN the flexibility to retain task-relevant information over different horizons, which the authors show improves performance over prior plastic and non-plastic RNNs on memory and meta-learning tasks (Rodriguez et al., 2022).

4 Grounding Plasticity in Neuron Dynamics

As discussed in Section 2.1, biological evidence suggests that synaptic plasticity is a complex mechanism involving both local temporal relationships between neurons and global oscillatory dynamics (Wespatat et al., 2004; Jutras & Buffalo, 2010; Einarsson et al., 2017). The plastic neural networks reviewed in Section 3 capture only a part of this complexity. In these methods, plasticity is driven primarily by pairwise correlations between pre- and post-synaptic activations. Eligibility traces in retroactive backpropamine (Miconi et al., 2020) provide a limited form of extended temporal context, but synchronization and population-level temporal structure remain largely unexplored in plastic deep neural networks.

In this work, we explore whether a plastic RNN can be grounded in extended temporal dynamics and synchronization, rather than instantaneous correlations alone. Doing so requires an underlying architecture whose representations already reflect these properties. The CTM (Darlow et al., 2025), a recently proposed

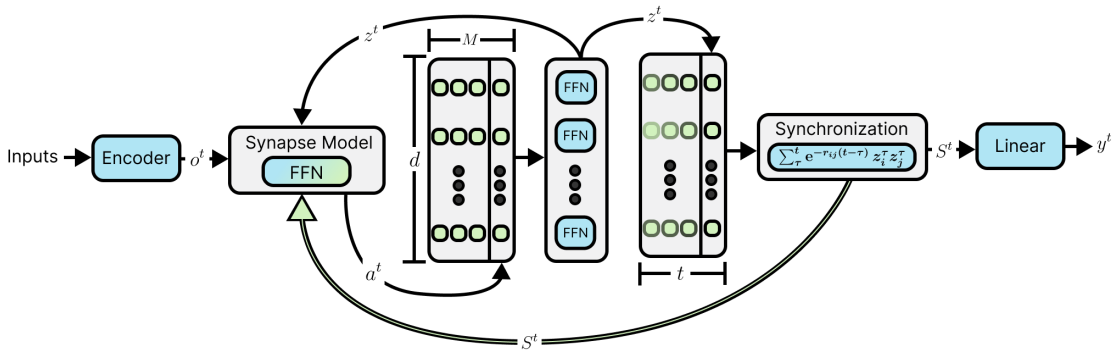


Figure 1: The Plastic CTM architecture. At each timestep t, the feedforward synapse model processes an observation and the previous activated state, the neuron-level models map past activations to the next activation, and the synchronization module computes time-decayed outer products of activation histories before projecting to output logits. The Plastic CTM uses synchronization to adjust the synapse weights.

bio-inspired RNN, exhibits exactly these characteristics. Each neuron in the CTM processes its own activation history through a unique set of learned weights, and the model uses pairwise neural synchronization as its latent representation. A natural next step is to investigate how plasticity can be introduced into the CTM. We start by reviewing the CTM in detail before introducing the Plastic CTM.

4.1 Continuous Thought Machines

The CTM (Darlow et al., 2025) combines two innovations well-suited to grounding plasticity in extended temporal dynamics, namely neuron-level temporal processing where each neuron processes its own activation history with a unique set of weights, and neural synchronization as the latent representation. We describe each in turn. More specifically, the CTM processes information recurrently in the following way. At each timestep t, the CTM takes as input the concatenation of the model's activated state $z^t \in \mathbb{R}^d$ and an external observation o^t, where d is the hidden size of the model. This input is fed through the CTM's feedforward *synapse* model, producing a pre-activated state a^t:

$$a^t = f_{\theta_{\text{syn}}}(z^t, o^t) \in \mathbb{R}^d, \tag{8}$$

where $f_{\theta_{\text{syn}}}$ is a learned feedforward network. This pre-activated state is appended to a sliding window of recent pre-activations, forming a matrix $A^t \in \mathbb{R}^{d \times M}$, where M is the window size. To produce the next activated state, each neuron i processes its own row of A^t through an independent *neuron-level model* (NLM):

$$z_i^{t+1} = g_{\theta_i}(A_i^t), \tag{9}$$

where g_{θ_i} is a small feedforward network with neuron-specific parameters θ_i. These NLMs allow the CTM to learn neuron-dependent mappings from the M most recent pre-activated states to the next activated state. Synchronization, the latent representation used to make predictions and interact with data via cross attention, is calculated via an exponentially weighted dot-product of activated state histories:

$$S_{ij}^t \;=\; \frac{\sum_{\tau=1}^{t} \mathrm{e}^{-r_{ij}(t-\tau)}\, z_i^\tau\, z_j^\tau}{\left(\sum_{\tau=1}^{t} \mathrm{e}^{-r_{ij}(t-\tau)}\right)^{1/2}}, \tag{10}$$

where r_{ij} is a learned decay parameter for each neuron pair, controlling how much past activated states contribute to the current synchronization value. The output of the model at each timestep then comes from a linear projection from the flattened synchronization matrix.

4.2 Plastic Continuous Thought Machines

Inspired by the biological motivation that population-level dynamics shape plasticity (Section 2.1), we propose the Plastic CTM, where the synchronization of NLM outputs modulates the synapse weights. Specifi-

cally, we replace the CTM synapse model in equation 8 with a plastic variant:

$$a^t = f_{\theta_{\text{syn}}}(z^t, o^t, S^t) = G^t(S^t)\, W_{\text{down}}\, \text{concat}(z^t, o^t) \tag{11}$$

where $W_{\text{down}} \in \mathbb{R}^{d \times (d + d_{\text{obs}})}$ projects the input to the model dimension d, and $G^t(S^t) \in \mathbb{R}^{d \times d}$ are the effective weights at time t, combining a fixed and plastic component, similarly to Equation (1).

The effective weights for the Plastic CTM combine fixed and plastic components,

$$G^t = \frac{W + P^t}{\|W + P^t\|_F}, \qquad P^t = \alpha^t S^t, \tag{12}$$

where $\alpha^t \in \mathbb{R}$ is a plasticity coefficient controlling the relative contribution of the plastic component, and the Frobenius norm $\| \cdot \|_F$ normalization stabilizes training—a known challenge in plastic RNNs (Miconi et al., 2018). Unlike prior works where the plasticity coefficient is a learned parameter held fixed at inference (Equations (2) and (7)), we let α^t adapt over time with the magnitude of the synchronization by introducing a learned scalar $\beta \in \mathbb{R}$ and setting $\alpha^t = \beta \|W\|_F / \|S^t\|_F$, so that β directly controls the ratio of plastic to fixed components, independent of the current scale of S^t.

A consequence of this design is that the Plastic CTM departs structurally from traditional Hebbian-style updates in two ways. First, the signal driving plasticity is the pairwise synchronization S^t, which captures correlations between all neuron pairs at the same timestep, rather than the outer product of pre- and post-synaptic activations at successive timesteps. Second, the activations z^τ entering S^t are themselves the outputs of temporal NLMs, so plasticity is grounded in extended firing histories rather than instantaneous correlations.

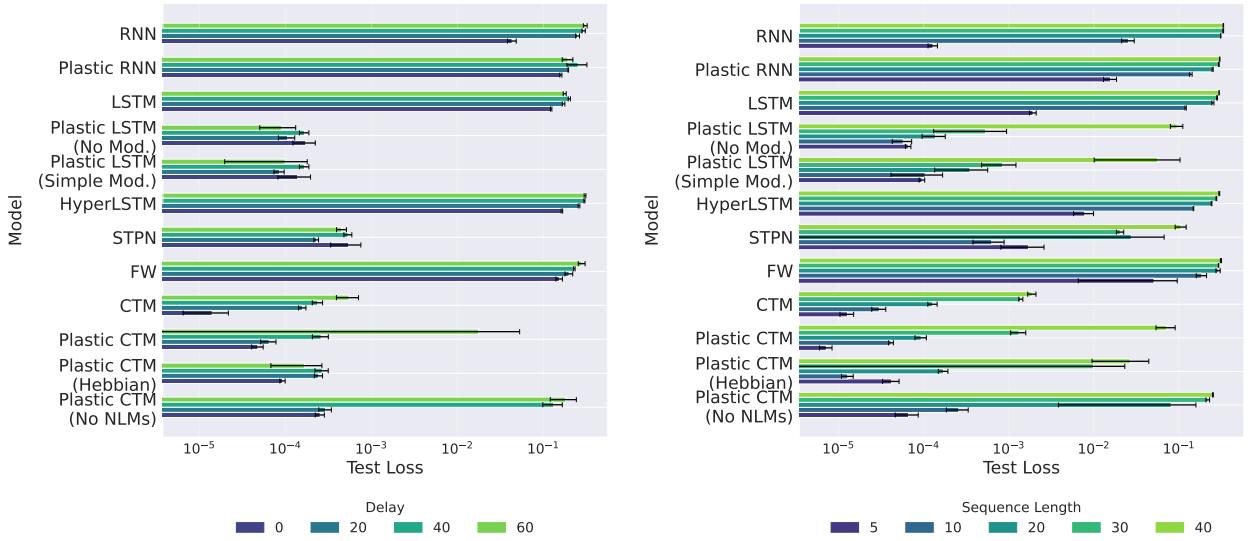
5 Experiments

We conduct a systematic comparison of plastic and non-plastic RNNs across four task families that are typically used in the literature. These cover algorithmic recall (Ba et al., 2016a; Rodriguez et al., 2022; Duan et al., 2023), in-context learning (Duan et al., 2023), continual learning (Miconi, 2016; Irie et al., 2025), and meta-RL (Miconi et al., 2018; 2020; Najarro & Risi, 2020; Rodriguez et al., 2022). Across these tasks we evaluate ten architectures, including the Plastic CTM, under parameter-matched conditions with per-task hyperparameter tuning. Beyond benchmark performance, we probe how each architecture actually uses its plasticity at inference time, and analyze the weight and hidden-state dynamics. Ablations on the Plastic CTM isolate the contributions of neuron-level temporal processing and synchronization-based updates.

5.1 Baselines and Model Ablations

Our evaluation covers ten architectures. The non-plastic baselines are a simple RNN with tanh activation, the LSTM (Hochreiter & Schmidhuber, 1997; Gers et al., 2000), and the CTM (Darlow et al., 2025). The plastic baselines are the methods surveyed in Section 3. These include the Plastic RNN (Miconi et al., 2018), the Plastic LSTM (Miconi et al., 2020) in two variants ("No Mod." with a learned scalar in place of M^t, and "Simple Mod." with M^t as a learned projection of the hidden state), and STPN (Rodriguez et al., 2022), together with two non-Hebbian rapid-adaptation architectures, an RNN with fast weights (FW; Ba et al., 2016a) and the HyperLSTM (Ha et al., 2016). All models use layer normalization (Ba et al., 2016b), are parameter-matched by adjusting hidden dimensions, and are tuned independently per task with Optuna (Akiba et al., 2019). CTM-based models use memory length $M = 10$ unless specified.

We additionally ablate the Plastic CTM in two ways. "No NLMs" removes NLMs so that synchronization is computed directly from pre-activated states a^t, isolating the contribution of per-neuron temporal processing. "Hebbian" replaces synchronization-based plasticity with the differentiable Hebbian rule of Miconi et al. (2018) on $a^{t-1} \otimes a^t$, scaled by a learned per-synapse α as in Miconi et al. (2018), instead of the time-varying α^t. NLMs and normalization of G^t are retained in the ablations. Diagrams of both ablations are given in Section A.



(a) Increasing delay m with fixed sequence length $n = 5$. (b) Fixed delay $m = 0$ with increasing sequence length n.

Figure 2: Test loss with varying delay m and sequence length n. Lower is better. A subset of architectures performs notably better, but the plastic/non-plastic distinction does not strictly separate them.

5.2 Memory and Recall: Copy Task

We evaluate memorization and recall using a sequential copying task adapted from Duan et al. (2023). In this task the model receives a sequence of n vectors in $\mathbb{R}^l$ with values sampled uniformly from $[-1, 1]$, followed by m blank timesteps, and must then reproduce the original sequence. Each input is $(l + 2)$-dimensional, with two binary flags indicating the input and recall phases, and performance is measured by mean squared error (MSE). We run two experiments: varying delay $m \in \{0, 20, 40, 60\}$ with fixed $n = 10$ to test retention, and varying length $n \in \{5, 10, 20, 30, 40\}$ with $m = 0$ to test capacity. We use $l = 10$ and parameter-match all models to approximately 12K parameters; LSTM-based models use chrono initialization (Tallec & Ollivier, 2018). Full details are given in Section C.

Results are shown in Figure 2. The top performers are the Plastic LSTM (with and without neuromodulation), STPN, and the Plastic CTM, along with the non-plastic CTM and the Hebbian ablation, while the RNN, Plastic RNN, LSTM, FW, and HyperLSTM perform notably worse. Plasticity alone does not explain these results. The Plastic RNN performs no better than the non-plastic RNN, and neuromodulation provides no benefit, with the two Plastic LSTM variants performing similarly.

The Plastic CTM ablations show that removing NLMs causes substantial degradation across both experiments, indicating that NLMs are the main source of memorization ability by giving each neuron direct access to its activation history. The Hebbian ablation, however, matches the full Plastic CTM, indicating that synchronization-based plasticity provides no advantage over standard Hebbian updates on this task.

5.3 One-Shot Adaptation: One-Shot Image Classification

We evaluate each architecture on a one-shot image classification task, a canonical test of rapid in-context learning (Vinyals et al., 2016; Finn et al., 2017). Following Finn et al. (2017); Duan et al. (2023), we conduct five-way one-shot classification experiments on the CIFAR-100 (Krizhevsky et al., 2009) and MiniImageNet (Ravi & Larochelle, 2017) datasets. Each trial consists of 10 image-label pairs presented sequentially. During the training phase (first 5 images), the model observes both images and labels to learn class associations. In the testing phase (final 5 images), labels are withheld and the model must infer the correct class. Following Duan et al. (2023), we use a 4-layer convolutional neural network to process input images before feeding them to the recurrent models. Full experimental details are provided in Section D.

Table 2: 5-way 1-shot classification accuracy on CIFAR-100 and Mini-ImageNet, showing the first test image accuracy (Acc.) and mean across all five test images (Acc. All). Results are averaged over 3 seeds with 95% CIs. MAML and ProtoNet results are from Duan et al. (2023). Grounding plasticity in extended temporal dynamics via the Plastic CTM shows noticeable improvements in performance.

	CIFAR-100		MiniImageNet	
Architecture/Algorithm	Acc.	Acc. All	Acc.	Acc. All
MAML (Finn et al., 2017)	58.9±1.9	-	48.7±1.8	-
ProtoNet (Snell et al., 2017)	55.5±0.7	-	53.5±0.6	-
RNN	45.9±1.8	46.8±1.4	42.7±0.7	43.3±0.9
Plastic RNN (Miconi et al., 2018)	45.9±1.3	46.4±1.1	43.8±1.8	44.3±2.1
LSTM (Hochreiter & Schmidhuber, 1997)	53.2±1.1	55.7±0.7	48.1±2.1	49.9±1.8
Plastic LSTM (No Mod.) (Miconi et al., 2020)	52.5±2.8	54.9±3.1	48.4±1.1	49.6±0.5
Plastic LSTM (Simple Mod.)	53.1±2.5	55.8±1.6	48.5±0.8	49.9±0.3
HyperLSTM (Ha et al., 2016)	49.9±2.3	52.3±2.5	45.6±0.9	46.6±2.0
STPN (Rodriguez et al., 2022)	56.5±3.2	58.8±2.6	**50.6±1.3**	**52.5±0.5**
FW (Ba et al., 2016a)	56.5±2.1	58.6±1.7	47.6±2.2	48.7±2.1
CTM (Darlow et al., 2025)	48.6±1.6	49.7±1.9	44.6±1.3	45.5±1.5
Plastic CTM (No NLMs)	56.6±3.2	59.3±2.1	49.4±2.5	51.5±2.4
Plastic CTM (Hebbian)	53.7±2.8	56.5±2.8	48.3±1.9	50.1±1.7
Plastic CTM (Ours)	**60.0±2.0**	**62.7±1.9**	50.3±1.6	**52.5±1.1**

Results are shown in Table 2. On CIFAR-100, the Plastic CTM achieves the highest accuracy at 60.0% on the first test image and 62.7% across all test images, outperforming MAML (58.9%) and ProtoNets (55.5%). On MiniImageNet, the Plastic CTM and STPN achieve comparable performance (50.3% and 50.6%), both above MAML (48.7%) but below ProtoNets (53.5%). The Plastic CTM substantially outperforms the non-plastic CTM, improving mean accuracy by 11.4% on CIFAR-100 and 5.7% on MiniImageNet, and consistently outperforms the differentiable Hebbian baselines, suggesting that grounding plasticity in temporally-processed activation histories provides an advantage over instantaneous Hebbian correlations in this task.

The ablations isolate where the Plastic CTM's advantage comes from. Removing NLMs reduces accuracy by 1–3%, indicating a modest contribution from per-neuron temporal processing. Replacing synchronization-based updates with standard Hebbian plasticity causes a larger degradation with substantially increased variance, in contrast to the copy task where the Hebbian ablation matched the full Plastic CTM.

5.4 In-Context Learning: Few-Shot Regression

We evaluate in-context function learning on a few-shot regression task commonly used to benchmark meta-learning methods (Finn et al., 2017; Duan et al., 2023). In each episode a new target mapping $f : [-1, 1]^n \to \mathbb{R}$ is sampled from one of two function families, a linear function or a small MLP. The model is presented with K noisy support pairs $(\mathbf{x}_t, f(\mathbf{x}_t) + \epsilon_t)$ with $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$, followed by query inputs whose outputs it must predict. Since the target function changes between episodes, the model cannot memorise any single mapping during training and must instead learn an in-context inference procedure. We use input dimension $n = 12$ for the Linear family and $n = 6$ for the MLP family, and report results for $K \in \{10, 20\}$. Full experimental details are provided in Section E.

Results are shown in Table 3. The Plastic CTM achieves the lowest test loss on both function families at both support sizes, with the Hebbian ablation of the Plastic CTM the closest competitor. The Plastic LSTM variants follow closely behind, with the remaining baselines performing notably worse. As in the one-shot classification setting, plastic architectures dominate, consistent with the view that in-context function learning benefits from explicit weight adaptation.

Table 3: Few-shot regression test loss (MSE) on Linear and MLP function learning tasks with K=20 and K=10 support examples. Results are averaged over 3 seeds with 95% CIs. The Plastic CTM outperforms all baselines, suggesting that grounding plasticity in extended dynamics is beneficial for in-context learning.

	Linear		MLP	
Architecture	K=20	K=10	K=20	K=10
RNN	0.339±0.010	0.483±0.006	0.067±0.026	0.166±0.007
Plastic RNN (Miconi et al., 2018)	0.268±0.228	0.361±0.004	0.053±0.010	0.140±0.003
LSTM (Hochreiter & Schmidhuber, 1997)	0.104±0.016	0.325±0.052	0.018±0.001	0.078±0.013
Plastic LSTM (No Mod.) (Miconi et al., 2020)	0.022±0.002	0.209±0.007	0.015±0.003	0.044±0.001
Plastic LSTM (Simple Mod.)	0.022±0.002	0.210±0.030	0.014±0.001	0.042±0.002
HyperLSTM (Ha et al., 2016)	0.099±0.002	0.289±0.005	0.026±0.002	0.092±0.026
STPN (Rodriguez et al., 2022)	0.090±0.023	0.274±0.006	0.021±0.001	0.093±0.002
FW (Ba et al., 2016a)	0.299±0.310	0.338±0.003	0.037±0.004	0.108±0.006
CTM (Darlow et al., 2025)	0.328±0.159	0.431±0.068	0.028±0.002	0.103±0.008
Plastic CTM (No NLMs)	0.367±0.035	0.498±0.031	0.358±0.086	0.233±0.006
Plastic CTM (Hebbian)	0.025±0.000	0.217±0.007	0.013±0.000	0.041±0.002
Plastic CTM (Ours)	**0.016±0.000**	**0.176±0.002**	**0.012±0.001**	**0.032±0.002**

5.5 Meta-Reinforcement Learning: Maze Exploration

Meta-RL is thought to be a key application of plasticity (Miconi et al., 2018; 2020; Rodriguez et al., 2022). We evaluate on the maze exploration task from Miconi et al. (2018), where an agent with only a local 3×3 egocentric view must navigate to a hidden goal. The goal remains fixed within each episode but varies across episodes; upon reaching it, the agent receives a large reward and is teleported to a random location. This tests the agent's ability to exploit discovered goal locations through memory-driven meta-RL (Wang et al., 2016; Duan et al., 2016). We trained all agents using Proximal Policy Optimization (PPO; Schulman et al., 2017) with Generalized Advantage Estimation (GAE; Schulman et al., 2015) and Pop-Art normalization (Van Hasselt et al., 2016). Following Miconi et al. (2020), all models were parameter-matched to an RNN with a hidden size of 100. Full experimental details are provided in Section F.

Results for each architecture are shown in Figure 3. Excluding FW, all architectures achieve a similar final reward, showing that the Plastic CTM matches earlier methods but also highlighting that this canonical task may not effectively discriminate between plastic networks. Notably, non-plastic architectures with larger hidden sizes (given parameter-matching) perform comparably to their plastic counterparts, often with qualitatively better training stability. We attribute this to modernized training, in particular PPO and Pop-Art in place of A2C (Mnih et al., 2016), under which non-plastic models can store sufficient task-

Figure 3: Average returns on the maze exploration task, with shaded areas indicating standard deviation calculated over three seeds. All models, except FW, perform similarly and learn to solve the task.

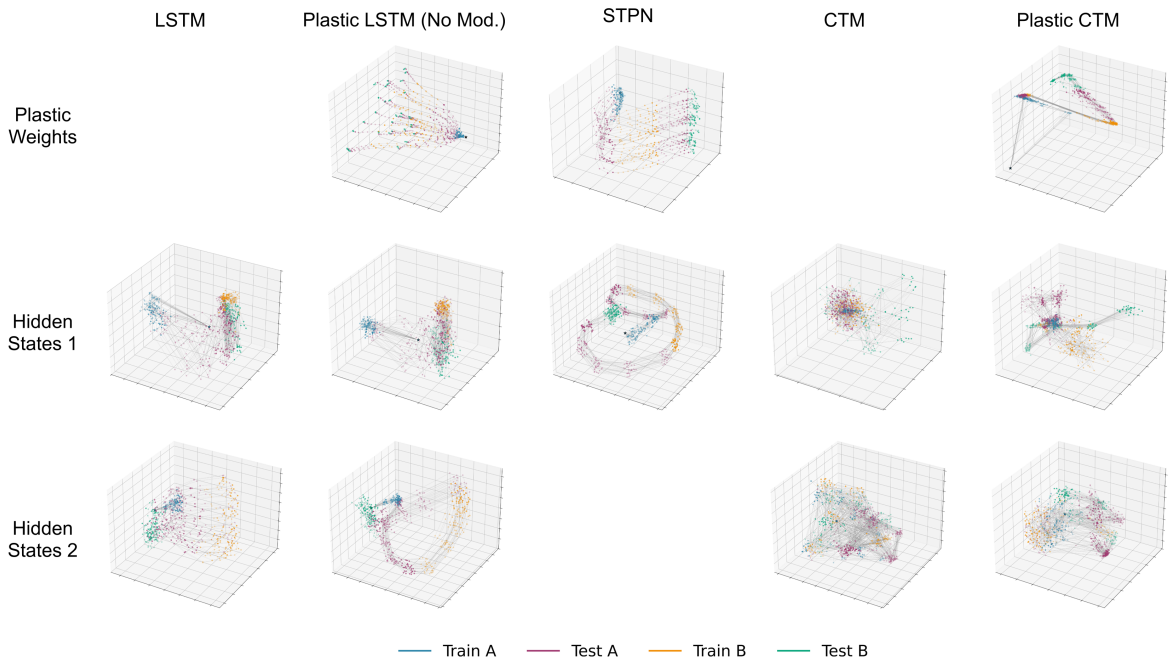


Figure 4: Weight and hidden state dynamics visualized via PCA; hidden states 1 and 2 refer to the hidden and cell states of the LSTM-based models, the hidden state for the STPN, and z^t and a^t for the CTM-based models. The plastic weights of the STPN and Plastic LSTM show more continual evolution, whilst the Plastic CTM's weights exhibit discrete phase-dependent jumps. The LSTM and Plastic LSTM hidden states follow similar dynamics, suggesting they solve the task in the same fashion.

specific information in their hidden states. A more challenging variant should require more episode-specific adaptation beyond memorizing a single goal.

5.6 Analysis: Continual Learning on Sequential Split-MNIST

Continual learning requires acquiring new skills without catastrophic forgetting (McCloskey & Cohen, 1989; Ratcliff, 1990; McClelland et al., 1995; French, 1999). Meta-learning approaches address this by learning forgetting-robust strategies (Beaulieu et al., 2020; Irie et al., 2025). We evaluate whether plastic architectures are amenable substrates for such strategies, while also using the setting to probe how each architecture uses its plasticity.

We use a modified version of the sequential Split-MNIST task from Irie et al. (2025). The dataset is split into Set A (digits 0–4) and Set B (digits 5–9), both randomly mapped to labels 0–4 per episode. Each episode presents Set A exemplars, Set A queries, Set B exemplars, Set A queries (testing retention), and Set B queries (testing forward transfer). During query phases, labels are withheld and the model must predict from in-context associations. Full experimental details are in Section G.

We find that both plastic and non-plastic versions of the LSTM and CTM, as well as the STPN, achieve near-perfect accuracy across all three query phases, demonstrating that plastic architectures can successfully meta-learn in-context continual learning algorithms. Notably, the non-plastic LSTM and CTM also reach near-perfect accuracy, suggesting that plasticity is not strictly necessary for the task and raising the question of whether the plastic architectures rely on plasticity at all. Earlier experiments already hint at different strategies. On the copy task the Plastic LSTM significantly outperforms the LSTM, yet on one-shot classification the two perform similarly, suggesting the Plastic LSTM uses plasticity primarily as additional memory rather than for task-specific adaptation. The Plastic CTM shows the opposite pattern, with substantial gains on one-shot classification and more modest gains on pure memorization.

Table 4: Accuracy change with plasticity disabled. The STPN and Plastic CTM rely on plasticity for all tasks, while the Plastic LSTM only shows a significant change for continual learning Set B forward accuracy.

	One-Shot Classification		Continual Learning		
Architecture	CIFAR-100	MiniImageNet	Set A Acc.	Set A Ret.	Set B Acc.
Plastic LSTM (No Mod.)	↑0.7%	↑1.1%	0.0%	↓3.1%	↓48.2%
STPN	↓65.9%	↓48.8%	↓79.1%	↓80.0%	↓79.9%
Plastic CTM (Ours)	↓66.6%	↓42.0%	↓68.4%	↓69.6%	↓77.4%

To test this directly, we disable plasticity at inference time for the top-performing plastic models and measure the drop in accuracy. The results in Table 4 confirm our interpretation. The STPN and Plastic CTM suffer dramatic drops when plasticity is disabled, indicating genuine reliance on plastic weights for in-context adaptation. The Plastic LSTM shows no degradation on one-shot learning and only loses forward accuracy on Set B in continual learning, consistent with plasticity serving primarily as auxiliary memory.

Following the neuroscience literature (Saxena & Cunningham, 2019), we further visualize these mechanistic differences by projecting plastic weights and hidden states into three dimensions using Principal Component Analysis (PCA) with Procrustes alignment (Gower, 1975) of equivalent hidden states (hidden/cell states for the LSTM variants, z^t/a^t for the CTM variants) for cross-model comparison (Figure 4). The Plastic LSTM and STPN exhibit smooth, continuous weight dynamics across task phases, while the Plastic CTM maintains discrete, well-separated weight configurations for each task phase, a pattern that may reduce inter-task interference. Hidden state dynamics reveal a similar distinction. The LSTM hidden and cell states show similar trajectories regardless of plasticity, while the Plastic CTM's activated states z^t exhibit markedly more clustered structure than its non-plastic counterpart. Quantitatively, the Procrustes disparity (0 for identical, $\sqrt{2}$ for orthogonal) between the CTM and the Plastic CTM's hidden states is 0.83–0.89, versus 0.42–0.45 for the LSTM variants. This suggests that the CTM and the Plastic CTM employ fundamentally different computational strategies, while both the LSTM and Plastic LSTM solve the task in largely similar ways.

6 Discussion and Conclusion

Synaptic plasticity underlies learning and memory in biological brains, and translating this mechanism into artificial neural networks offers a promising path toward more adaptive AI systems. This paper describes existing differentiable plastic RNNs under a unified notation, presents a systematic parameter-matched evaluation of plastic RNNs, and identifies grounding plasticity in neuron dynamics and synchronization as an underexplored design direction, which we instantiate with the Plastic CTM. Our findings highlight that grounding plasticity in neuron dynamics is effective for in-context adaptation, that not all plastic architectures actually use their plasticity, that those that do can adopt fundamentally different weight and hidden-state dynamics, and that the canonical maze meta-RL benchmark is saturated under modern training algorithms. Benchmark accuracy alone is therefore insufficient for evaluating plastic RNNs. The field should pair benchmark results with mechanistic analysis, develop more discriminating benchmarks, and explore richer biologically inspired rules such as population-level synchronization.

Like other plastic RNNs, the Plastic CTM incurs substantial per-step compute and memory overhead, with its forward and backward pass roughly four times slower than a non-plastic LSTM. More efficient parameterizations of plastic weights remain an active research direction. Full per-architecture timings and memory costs are reported in Section H. Plastic RNNs can also exhibit training instability, as evidenced by the Plastic CTM on the meta-RL task and the elevated seed-to-seed variance on one-shot image classification task. Characterizing the sources of this instability and developing principled remedies is an important direction for future work. Finally, our results show that performance on the meta-RL maze task is saturated across recent methods, limiting its ability to discriminate between approaches. Progress will require more demanding variants, such as randomizing the maze structure across episodes, which would require models to acquire a world model at inference time rather than amortize it through the outer optimization loop. Developing plastic RNNs for harder meta-RL tasks remains an open challenge.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. *Advances in neural information processing systems*, 29, 2016a.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016b.

Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. *arXiv preprint arXiv:2002.09571*, 2020.

Saptarshi Bej, Gouri Lakshmi, Harshit Kumar, Pritam Kar, Bikas C Das, et al. Spike agreement dependent plasticity: A scalable bio-inspired learning paradigm for spiking neural networks. *arXiv preprint arXiv:2508.16216*, 2025.

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Optimality in Biological and Artificial Networks?*, pp. 265–287. Routledge, 2013.

Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience*, 18(24):10464–10472, 1998.

Paolo Calabresi, Barbara Picconi, Alessandro Tozzi, and Massimiliano Di Filippo. Dopamine-mediated regulation of corticostriatal synaptic plasticity. *Trends in neurosciences*, 30(5):211–219, 2007.

Athira Chandrasekharan, Harshit Kumar, Bikas C Das, Saptarshi Bej, et al. Supervised spike agreement dependent plasticity for fast local learning in spiking neural networks. *arXiv preprint arXiv:2601.08526*, 2026.

Luke Darlow, Ciaran Regan, Sebastian Risi, Jeffrey Seely, and Llion Jones. Continuous thought machines. *arXiv preprint arXiv:2505.05522*, 2025.

Rousslan Fernand Julien Dossa, Shengyi Huang, Santiago Ontañón, and Takashi Matsubara. An empirical investigation of early stopping optimizations in proximal policy optimization. *IEEE access*, 9:117981–117992, 2021.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Yu Duan, Zhongfan Jia, Qian Li, Yi Zhong, and Kaisheng Ma. Hebbian and gradient-based plasticity enables robust memory and rapid learning in rnns. *arXiv preprint arXiv:2302.03235*, 2023.

Hafsteinn Einarsson, Marcelo M Gauy, Johannes Lengler, and Angelika Steger. A model of fast hebbian spike latency normalization. *Frontiers in computational neuroscience*, 11:33, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4): 128–135, 1999.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

John C Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Donald Olding Hebb. *The organization of behavior: A neuropsychological theory.* Psychology press, 2005.

Geoffrey E Hinton and David C Plaut. Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the Cognitive Science Society*, pp. 177–186, 1987.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. Metalearning continual learning algorithms. *Transactions on Machine Learning Research*, 2025.

Michael J Jutras and Elizabeth A Buffalo. Synchronous neural activity and memory formation. *Current opinion in neurobiology*, 20(2):150–155, 2010.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Louis Kirsch and Jürgen Schmidhuber. Meta learning backpropagation and improving it. *Advances in Neural Information Processing Systems*, 34:14122–14134, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Binggwong Leung, Worasuchad Haomachai, Joachim Winther Pedersen, Sebastian Risi, and Poramate Manoonpong. Bio-inspired plastic neural networks for zero-shot out-of-distribution generalization in complex animal-inspired robots. *arXiv preprint arXiv:2503.12406*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Stephen J Martin, Paul D Grimwood, and Richard GM Morris. Synaptic plasticity and memory: an evaluation of the hypothesis. *Annual review of neuroscience*, 23(1):649–711, 2000.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Thomas Miconi. Learning to learn with backpropagation of hebbian plasticity. *arXiv preprint arXiv:1609.02228*, 2016.

Thomas Miconi, Kenneth Stanley, and Jeff Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *International Conference on Machine Learning*, pp. 3559–3568. PMLR, 2018.

Thomas Miconi, Aditya Rawal, Jeff Clune, and Kenneth O Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. *arXiv preprint arXiv:2002.10585*, 2020.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.

Tsendsuren Munkhdalai and Adam Trischler. Metalearning with hebbian fast weights. *arXiv preprint arXiv:1807.05076*, 2018.

Elias Najarro and Sebastian Risi. Meta-learning through hebbian plasticity in random networks. *Advances in Neural Information Processing Systems*, 33:20719–20731, 2020.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15 (3):267–273, 1982.

Joachim Pedersen, Erwan Plantec, Eleni Nisioti, Milton Montero, and Sebastian Risi. Structurally flexible neural networks: Evolving the building blocks for general agents. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1119–1127, 2024.

Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2017.

Hector Garcia Rodriguez, Qinghai Guo, and Timoleon Moraitis. Short-term plasticity neurons learning to learn and forget. In *International Conference on Machine Learning*, pp. 18704–18722. PMLR, 2022.

Shreya Saxena and John P Cunningham. Towards the neural population doctrine. *Current opinion in neurobiology*, 55:103–111, 2019.

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

Jürgen Schmidhuber, Jieyu Zhao, and Marco Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28(1):105–130, 1997.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Andrea Soltoggio, Kenneth O Stanley, and Sebastian Risi. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *Neural Networks*, 108:48–67, 2018.

Corentin Tallec and Yann Ollivier. Can recurrent neural networks warp time? In *International Conference on Learning Representations*, 2018.

Yuchen Tian, Assel Kembay, Nhan Duy Truong, Jason K Eshraghian, and Omid Kavehei. Beyond pairwise plasticity: Group-level spike synchrony facilitates efficient learning in spiking neural networks. *arXiv preprint arXiv:2505.14841*, 2025.

Hado P Van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. *Advances in neural information processing systems*, 29, 2016.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

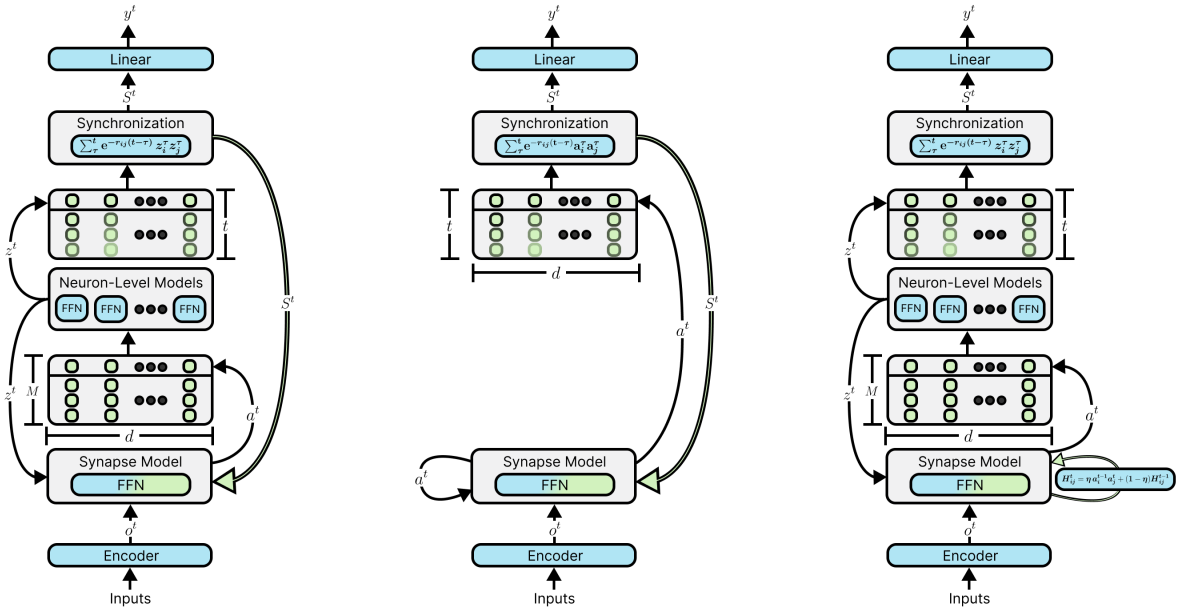
Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Valérie Wespatat, Frank Tennigkeit, and Wolf Singer. Phase sensitivity of synaptic modifications in oscillating cells of rat visual cortex. *Journal of Neuroscience*, 24(41):9067–9075, 2004.

Sho Yagishita, Akiko Hayashi-Takagi, Graham CR Ellis-Davies, Hidetoshi Urakubo, Shin Ishii, and Haruo Kasai. A critical time window for dopamine actions on the structural plasticity of dendritic spines. *Science*, 345(6204):1616–1620, 2014.

A Ablation Diagrams

We provide a more detailed overview of the ablated variants of the Plastic CTM in Figure 5.



(a) Plastic CTM (b) Plastic CTM (No NLMs) (c) Plastic CTM (Hebbian)

Figure 5: The Plastic CTM (a) and the two ablated variants. The No NLMs variant (b) computes synchronization from the output of the synapse model a^t, without neuron-level temporal processing. The Hebbian variant (c) computes plasticity using standard Hebbian plasticity rules based on the outer product of pre- and post-activations, as in Miconi et al. (2018).

B Hyperparameter Tuning

For the copy, one-shot image classification, and few-shot regression tasks, we tune learning rate and dropout independently per architecture (and per task condition, where applicable) using Optuna (Akiba et al., 2019). Each search runs for 20 trials with the multivariate Tree-structured Parzen Estimator (TPE) sampler. The first 10 trials are sampled uniformly at random to seed the TPE surrogate, and the remaining 10 trials are proposed by maximising expected improvement under the surrogate using 24 candidate samples per step. The objective is the best validation metric reached during training.

The default search ranges are learning rate in $[10^{-4}, 10^{-3}]$ and dropout in $[0.0, 0.5]$. For Fast Weights, each search additionally covers λ in $[0.8, 0.99]$, η in $[0.1, 0.9]$, and the number of inner loop steps in $\{1, 2, 3\}$. The few-shot regression task extends the learning rate range to $[10^{-4}, 10^{-2}]$ and does not tune dropout (dropout is fixed at 0.0 for all architectures).

The maze exploration and Sequential Split-MNIST tasks do not use Optuna, because the reported architectures reach perfect or near-perfect accuracy under shared, manually selected hyperparameters, making

further tuning unnecessary. Maze exploration uses a fixed learning rate of 3×10^{-4} for all architectures, following standard PPO settings, with FW-specific values listed in Section F. We did run an Optuna search for Fast Weights on maze exploration, but were unable to identify hyperparameters that performed on par with the other architectures, and so we report results under the manually selected values in Section F. For Sequential Split-MNIST, we report only the three architectures that achieved non-trivial performance under shared, manually selected hyperparameters (LR $= 10^{-4}$, dropout $= 0.2$, reported in Table 15). The remaining architectures failed to learn the task across the settings we tried.

The selected hyperparameters for each task are reported in the corresponding appendix section below.

C Experimental Details: Copy Task

C.1 Task Configuration and Architecture

We evaluate all models on a sequential copy task designed to test memorization and recall capabilities. The model receives a sequence of n vectors in $\mathbb{R}^l$ sampled uniformly from $[-1, 1]$, followed by m blank timesteps, and must then reproduce the original sequence in order. Each input is $(l + 2)$-dimensional, where the additional 2 dimensions correspond to binary flags indicating the input phase and recall phase, respectively. The model outputs the predictions sequentially, and performance is measured using the MSE:

$$\text{MSE} = \frac{1}{n} \sum_{t=1}^{n} \|\hat{y}^t - y^t\|^2, \tag{13}$$

where $\hat{y}_t$ are the predictions during the recall phase and y_t are the target values.

We conduct experiments varying two key parameters. First, we vary the delay $m \in \{0, 20, 40, 60\}$ with fixed sequence length $n = 10$ to evaluate retention over time. Second, we vary the sequence length $n \in \{5, 10, 20, 30, 40\}$ with fixed delay $m = 0$ to evaluate scaling to longer sequences. Both experiments use a fixed vector size of $l = 10$.

C.2 Model Configurations and Hyperparameters

All models are parameter-matched to approximately 12K parameters by adjusting their hidden dimensions d. Table 5 shows the specific configurations used for each architecture. The Plastic LSTM variants use either Hebbian plasticity without neuromodulation or with simple neuromodulation as described in Miconi et al. (2020). The HyperLSTM uses a hypernetwork of size 8, with embedding dimension 8. All experiments use a copy dimension of 10.

For each sweep condition, we perform a separate hyperparameter search following the protocol described in Section B. Tables 6–7 show the resulting hyperparameters per delay condition, and Tables 8–9 show the resulting hyperparameters per sequence length condition. Table 10 shows the FW-specific hyperparameters.

Table 5: Model configurations for copy task experiments.

Model	d	Params
RNN	98	12,162
Plastic RNN	72	12,251
LSTM	46	12,246
Plastic LSTM (No Mod.)	42	12,275
Plastic LSTM (Simple Mod.)	41	11,901
HyperLSTM	32	12,170
STPN	56	12,218
FW	99	12,385
CTM	65	12,230
Plastic CTM (No NLMs)	53	12,625
Plastic CTM (Hebbian)	55	12,276
Plastic CTM (Ours)	49	12,359

Table 6: Optimal hyperparameters for delay sweep: delay $\in \{0, 20\}$ (sequence length=10).

Model	Delay= 0		Delay= 20	
	Dropout	LR	Dropout	LR
RNN	0.001	2.53×10^{-4}	0.235	6.96×10^{-4}
Plastic RNN	0.002	9.51×10^{-4}	0.005	9.34×10^{-4}
LSTM	0.023	6.65×10^{-4}	0.023	9.67×10^{-4}
Plastic LSTM (No Mod.)	0.014	7.74×10^{-4}	0.008	9.15×10^{-4}
Plastic LSTM (Simple Mod.)	0.009	7.16×10^{-4}	0.005	6.87×10^{-4}
HyperLSTM	0.352	6.52×10^{-4}	0.151	7.79×10^{-4}
STPN	0.019	9.87×10^{-4}	0.003	6.65×10^{-4}
FW	0.015	8.90×10^{-4}	0.021	9.28×10^{-4}
CTM	6.9×10^{-5}	7.32×10^{-4}	0.003	6.78×10^{-4}
Plastic CTM (No NLMs)	0.164	8.51×10^{-4}	0.065	9.80×10^{-4}
Plastic CTM (Hebbian)	0.007	4.27×10^{-4}	0.003	8.69×10^{-4}
Plastic CTM (Ours)	0.004	6.76×10^{-4}	3.3×10^{-4}	6.74×10^{-4}

Table 7: Optimal hyperparameters for delay sweep: delay $\in \{40, 60\}$ (sequence length=10).

Model	Delay= 40		Delay= 60	
	Dropout	LR	Dropout	LR
RNN	0.018	7.17×10^{-4}	0.008	4.98×10^{-4}
Plastic RNN	0.092	9.00×10^{-4}	0.051	7.64×10^{-4}
LSTM	0.326	9.83×10^{-4}	0.030	9.40×10^{-4}
Plastic LSTM (No Mod.)	0.018	5.53×10^{-4}	0.005	9.86×10^{-4}
Plastic LSTM (Simple Mod.)	0.012	8.12×10^{-4}	0.002	6.69×10^{-4}
HyperLSTM	0.389	6.87×10^{-4}	0.348	8.33×10^{-4}
STPN	0.012	9.34×10^{-4}	0.005	9.34×10^{-4}
FW	0.366	7.66×10^{-4}	0.323	9.72×10^{-4}
CTM	0.004	3.62×10^{-4}	0.006	6.86×10^{-4}
Plastic CTM (No NLMs)	0.430	9.64×10^{-4}	0.259	9.84×10^{-4}
Plastic CTM (Hebbian)	0.002	8.36×10^{-4}	7.9×10^{-4}	8.57×10^{-4}
Plastic CTM (Ours)	0.004	6.13×10^{-4}	0.001	4.10×10^{-4}

Table 8: Optimal hyperparameters for sequence length sweep: length $\in \{5, 10, 20\}$ (delay=0).

Model	Seq. Len.= 5		Seq. Len.= 10		Seq. Len.= 20	
	Dropout	LR	Dropout	LR	Dropout	LR
RNN	1.4×10^{-4}	2.82×10^{-4}	0.005	4.04×10^{-4}	0.296	9.79×10^{-4}
Plastic RNN	0.003	7.61×10^{-4}	0.010	7.06×10^{-4}	0.003	6.70×10^{-4}
LSTM	0.005	8.19×10^{-4}	0.006	7.12×10^{-4}	0.003	9.15×10^{-4}
Plastic LSTM (No Mod.)	0.005	9.29×10^{-4}	0.002	6.01×10^{-4}	0.008	7.89×10^{-4}
Plastic LSTM (Simple Mod.)	0.008	9.59×10^{-4}	0.003	9.97×10^{-4}	0.003	9.97×10^{-4}
HyperLSTM	0.225	9.70×10^{-4}	0.090	1.00×10^{-3}	0.140	9.72×10^{-4}
STPN	0.131	9.61×10^{-4}	0.022	8.35×10^{-4}	0.069	9.90×10^{-4}
FW	0.142	9.15×10^{-4}	0.156	7.67×10^{-4}	0.146	5.98×10^{-4}
CTM	0.001	4.19×10^{-4}	0.002	5.23×10^{-4}	0.004	3.87×10^{-4}
Plastic CTM (No NLMs)	0.113	9.65×10^{-4}	0.199	9.75×10^{-4}	0.023	9.45×10^{-4}
Plastic CTM (Hebbian)	0.006	6.32×10^{-4}	7.9×10^{-5}	9.03×10^{-4}	0.005	7.03×10^{-4}
Plastic CTM (Ours)	3.2×10^{-4}	3.63×10^{-4}	0.003	5.55×10^{-4}	0.002	8.19×10^{-4}

Table 9: Optimal hyperparameters for sequence length sweep: length $\in \{30, 40\}$ (delay=0).

Model	Seq. Len.= 30		Seq. Len.= 40	
	Dropout	LR	Dropout	LR
RNN	0.245	1.35×10^{-4}	0.241	7.32×10^{-4}
Plastic RNN	8.5×10^{-4}	5.06×10^{-4}	0.010	8.67×10^{-4}
LSTM	0.014	9.65×10^{-4}	0.033	9.70×10^{-4}
Plastic LSTM (No Mod.)	0.006	9.89×10^{-4}	0.034	8.43×10^{-4}
Plastic LSTM (Simple Mod.)	0.006	9.60×10^{-4}	0.063	9.49×10^{-4}
HyperLSTM	0.018	8.44×10^{-4}	0.038	9.18×10^{-4}
STPN	0.061	9.90×10^{-4}	0.089	9.82×10^{-4}
FW	0.009	8.61×10^{-4}	0.167	8.96×10^{-4}
CTM	0.017	6.17×10^{-4}	0.013	8.28×10^{-4}
Plastic CTM (No NLMs)	0.010	9.52×10^{-4}	0.006	9.65×10^{-4}
Plastic CTM (Hebbian)	0.024	8.62×10^{-4}	0.049	8.12×10^{-4}
Plastic CTM (Ours)	0.012	8.73×10^{-4}	0.057	9.10×10^{-4}

Table 10: FW-specific hyperparameters per condition.

Sweep	Condition	λ	η	Steps
Delay	0	0.904	0.161	1
Delay	20	0.870	0.258	1
Delay	40	0.965	0.306	1
Delay	60	0.915	0.601	1
Seq. Len.	5	0.974	0.880	1
Seq. Len.	10	0.923	0.324	1
Seq. Len.	20	0.886	0.221	1
Seq. Len.	30	0.863	0.122	2
Seq. Len.	40	0.910	0.376	1

C.3 Training and Evaluation

All models were trained for 1000 epochs with a batch size of 64 using the AdamW optimizer (Loshchilov & Hutter, 2017). We applied gradient clipping with a maximum norm of 5. The learning rate and dropout for each architecture were tuned as described in Section B, and the resulting values are reported in Tables 6–10. For each experimental condition, we trained 5 random seeds.

D Experimental Details: One-Shot Image Classification

D.1 Task Configuration and Architecture

We evaluate all models on 5-way 1-shot classification using CIFAR-100 and Mini-ImageNet. Following the protocol established in Duan et al. (2023), each episode consists of 10 sequential image-label pairs: 5 training images with labels (1 exemplar per class) followed by 5 test images without labels. All images are processed through a shared 4-layer convolutional encoder before being fed to the recurrent networks. This encoder follows the ProtoNet architecture (Snell et al., 2017) and consists of four blocks, each containing a 3×3 convolution (64 channels, stride 1, padding 1), batch normalization, ReLU activations, and 2×2 max pooling. The flattened convolutional features are projected to a 256-dimensional embedding, which is concatenated with the one-hot labels during the training phase, or zero-vectors during the testing phase, before being processed by the recurrent model.

D.2 Model Configurations and Hyperparameters

All models are parameter-matched by adjusting their hidden dimensions d. Table 11 shows the hidden dimensions used for each architecture, and the tuned hyperparameters for CIFAR-100 and MiniImageNet. Plastic LSTM includes versions with and without neuromodulation (Miconi et al., 2020). The Plastic RNN uses standard Hebbian learning rules. The HyperLSTM uses a hypernetwork of size 64 with embedding dimension 32. Learning rate and dropout are tuned independently for each model and dataset using the Optuna protocol described in Section B.

Table 11: Model dimensions and hyperparameters for one-shot classification experiments. All models use $d_{\text{input}} = 256$ with identical encoders. Models are parameter-matched to approximately 520k parameters for CIFAR-100 and 860k parameters for MiniImageNet. FW uses additional hyperparameters: λ=0.909, η=0.196, steps=1 (CIFAR-100); λ=0.953, η=0.140, steps=1 (MiniImageNet).

		CIFAR-100			MiniImageNet		
Model	d	Params	LR	Dropout	Params	LR	Dropout
RNN	464	518k	9.53×10^{-4}	0.011	862k	3.72×10^{-4}	0.028
Plastic RNN	350	517k	9.28×10^{-4}	7.9×10^{-4}	861k	5.40×10^{-4}	0.017
LSTM	188	519k	8.45×10^{-4}	0.217	863k	8.60×10^{-4}	0.477
Plastic LSTM (No Mod.)	175	517k	7.21×10^{-4}	0.357	861k	9.63×10^{-4}	0.465
Plastic LSTM (Simple Mod.)	175	518k	9.84×10^{-4}	0.412	862k	9.28×10^{-4}	0.320
HyperLSTM	110	521k	9.15×10^{-4}	0.288	865k	9.12×10^{-4}	0.352
STPN	230	517k	6.01×10^{-4}	0.140	861k	6.74×10^{-4}	0.497
FW	464	518k	5.45×10^{-4}	0.453	862k	7.57×10^{-4}	0.167
CTM	340	512k	7.76×10^{-4}	0.010	856k	9.64×10^{-4}	0.042
Plastic CTM (No NLMs)	260	519k	9.37×10^{-4}	0.097	863k	9.25×10^{-4}	0.152
Plastic CTM (Hebbian)	290	518k	7.73×10^{-4}	0.007	862k	7.78×10^{-4}	0.016
Plastic CTM (Ours)	256	517k	6.52×10^{-4}	0.368	861k	9.37×10^{-4}	0.199

D.3 Training and Evaluation

All models were trained for 40 epochs with a batch size of 512 using the Adam optimizer (Kingma, 2014). We applied gradient clipping with a maximum norm of 1. As in Duan et al. (2023), we report the accuracy as the classification accuracy on the first test image, and additionally report the combined accuracy across all five test images. Results are averaged over 3 random seeds, with standard error reported. All models used identical data splits and random seeds to ensure fair comparison.

E Experimental Details: Few-Shot Regression

E.1 Task Configuration

We evaluate in-context function learning on a few-shot regression task adapted from the meta-learning literature (Finn et al., 2017; Duan et al., 2023). In each episode, a target function $f : [-1, 1]^n \to \mathbb{R}$ is sampled from one of two families:

- **Linear**: $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ with $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ sampled from a standard normal distribution.

- **MLP**: a small two-hidden-layer MLP with random Gaussian weights and a ReLU non-linearity, mapping $\mathbf{x} \in \mathbb{R}^n$ to a scalar.

The model is presented with K noisy support pairs $(\mathbf{x}_t, f(\mathbf{x}_t) + \epsilon_t)$ with $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$, followed by query inputs whose outputs it must predict. Inputs are sampled uniformly from $[-1, 1]^n$. We use input dimension $n = 12$ for the Linear family and $n = 6$ for the MLP family, and evaluate two support sizes $K \in \{10, 20\}$. Episodes are generated on the fly, so the model cannot memorise any single mapping during training.

E.2 Model Configurations and Hyperparameters

All models share an identical input/output interface. At each support step, the model receives the concatenation $[\mathbf{x}_t, y_t]$, where $y_t = f(\mathbf{x}_t) + \epsilon_t$, and at query steps the target slot is masked. The recurrent network outputs a scalar prediction at each query step. Hidden dimensions are tuned to parameter-match the architectures to approximately 150K parameters, and learning rate is tuned per architecture and per K using the Optuna protocol described in Section B. Tables 12 and 13 report the resulting hyperparameters for the Linear and MLP function families respectively. For the CTM variants, we initialize the NLM weights so that each NLM begins training as the identity mapping, where the most recent pre-activation becomes the post-activation. We found this could improve performance for CTM variants in this task.

Table 12: Model dimensions and hyperparameters for the few-shot regression task on the Linear function family ($n = 12$). Models are parameter-matched to approximately 150K parameters by adjusting the hidden dimension d. FW uses additional hyperparameters: λ=0.954, η=0.805, steps=1 (K=10); λ=0.908, η=0.375, steps=2 (K=20).

		$K = 10$		$K = 20$	
Model	d	Params	LR	Params	LR
RNN	270	151,201	1.69×10^{-4}	151,201	2.58×10^{-4}
Plastic RNN	220	149,602	2.45×10^{-4}	149,602	5.41×10^{-3}
LSTM	135	151,201	9.42×10^{-3}	151,201	2.67×10^{-3}
Plastic LSTM (No Mod.)	128	152,578	1.13×10^{-3}	152,578	2.63×10^{-3}
Plastic LSTM (Simple Mod.)	128	152,962	1.98×10^{-3}	152,962	2.79×10^{-3}
HyperLSTM	72	150,529	2.06×10^{-3}	150,529	6.54×10^{-3}
STPN	156	149,137	5.70×10^{-4}	149,137	3.65×10^{-3}
FW	269	150,103	3.27×10^{-3}	150,103	6.34×10^{-3}
CTM	200	142,402	4.49×10^{-4}	148,402	9.52×10^{-4}
Plastic CTM (No NLMs)	220	150,042	1.17×10^{-4}	150,042	1.35×10^{-4}
Plastic CTM (Hebbian)	177	149,762	2.56×10^{-4}	150,452	3.25×10^{-4}
Plastic CTM (Ours)	160	145,602	5.66×10^{-4}	150,402	2.65×10^{-3}

Table 13: Model dimensions and hyperparameters for the few-shot regression task on the MLP function family ($n = 6$). Models are parameter-matched to approximately 150K parameters by adjusting the hidden dimension d. FW uses additional hyperparameters: λ=0.864, η=0.139, steps=1 (K=10); λ=0.954, η=0.487, steps=1 (K=20).

		$K = 10$		$K = 20$	
Model	d	Params	LR	Params	LR
RNN	270	149,581	2.90×10^{-4}	149,581	2.54×10^{-4}
Plastic RNN	220	148,282	8.99×10^{-4}	148,282	3.60×10^{-3}
LSTM	135	150,391	3.31×10^{-3}	150,391	5.04×10^{-3}
Plastic LSTM (No Mod.)	128	151,810	7.73×10^{-3}	151,810	5.30×10^{-3}
Plastic LSTM (Simple Mod.)	128	152,194	2.02×10^{-3}	152,194	8.45×10^{-3}
HyperLSTM	72	150,097	3.51×10^{-3}	150,097	4.76×10^{-3}
STPN	156	150,093	1.88×10^{-3}	150,093	1.22×10^{-3}
FW	269	148,489	1.31×10^{-3}	148,489	1.23×10^{-3}
CTM	200	141,202	2.97×10^{-4}	147,202	3.25×10^{-4}
Plastic CTM (No NLMs)	220	150,061	1.49×10^{-4}	150,061	3.69×10^{-3}
Plastic CTM (Hebbian)	177	150,232	2.87×10^{-4}	150,946	3.66×10^{-4}
Plastic CTM (Ours)	160	144,642	1.06×10^{-3}	149,442	2.20×10^{-3}

E.3 Training and Evaluation

Models are trained with the AdamW optimizer (Loshchilov & Hutter, 2017) on the mean squared error between the predicted and true query outputs. We apply gradient clipping with maximum norm 5. Test loss is reported as the MSE on held-out episodes, averaged over 3 random seeds with 95% confidence intervals.

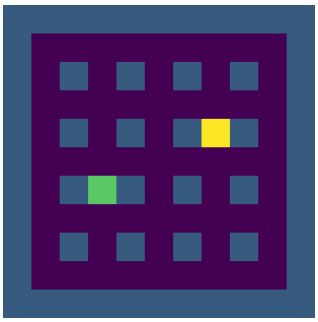


Figure 6: Maze layout for the maze exploration task. The yellow and green squares refer to the agent and the goal location, respectively. The light and dark blue areas correspond to walls and traversable locations, respectively. The agent's start location and the goal location are shuffled each episode, and the agent's location is set to a new random location each time the goal is reached.

F Experimental Details: Maze Exploration

F.1 Task Description and Environment

We evaluate all models on a maze exploration task following the protocol established in Miconi et al. (2018; 2020). The environment, shown in Figure 6, consists of an 11×11 grid-based maze with walls along the borders and pillars placed at even-numbered positions (2, 4, 6, 8) to create a structured navigation challenge. The agent receives only a local, egocentric 3×3 view of its surroundings, represented as a flattened 9-dimensional binary vector where 1 indicates a wall or pillar and 0 indicates free space.

At the start of each episode, both the agent and a goal location are randomly positioned in empty cells of the maze. The agent must navigate to the goal using four discrete actions: up, down, left, and right. Upon reaching the goal, the agent receives a reward of $+10$ and is immediately respawned at a new random location within the same episode, while the goal position remains fixed. If the agent attempts to move into a wall, it receives a penalty of -0.1 and remains in its current position. Each episode lasts 200 timesteps, during which the agent can potentially reach the goal multiple times.

This task design creates a meta-RL challenge where agents must rapidly adapt their behavior once they discover the goal location, transitioning from exploratory to exploitative navigation. Because the goal position varies across episodes but remains fixed within an episode, successful performance requires both learning general navigation strategies through the network's fixed parameters and rapid adaptation to memorize the goal location through plasticity mechanisms. The separation of fast plasticity-driven adaptation (remembering where the goal is) and slower parameter learning (learning how to navigate) is thought to make this an effective test of plastic networks' ability to perform temporal credit assignment in partially observable environments.

F.2 RL^2 Formulation

Following Miconi et al. (2018), we frame the maze exploration task as a meta-RL problem using the RL^2 algorithm (Duan et al., 2016; Wang et al., 2016). In this formulation, the recurrent network's hidden state serves as a memory that enables learning within an episode. To provide the network with information about the consequences of its actions, the observation space is augmented with the previous action and previous reward, creating an extended observation vector of dimension $9 + 4 + 1 = 14$, where 9 dimensions encode the agent and its surroundings, 4 dimensions encode the previous action as a one-hot vector, and 1 dimension encodes the previous reward.

At the start of each episode, the previous action is initialized to a zero vector and the previous reward is initialized to 0. After each step, the observation is augmented with the action taken and reward received at that step before being fed to the network at the next timestep. This allows the recurrent network to learn to

associate state-action pairs with outcomes and adjust its policy accordingly within a single episode, without requiring explicit weight updates during inference.

F.3 Model Configurations and Hyperparameters

All models are parameter-matched to approximately 12.3K total parameters by adjusting their hidden dimensions d. Table 14 shows the specific configurations used for each architecture. The HyperLSTM uses a hidden size of 16 for the hypernetwork and an embedding size of 4. The FW model uses a decay rate $\lambda = 0.9$, learning rate $\eta = 0.5$, and 1 inner loop step.

Table 14: Model configurations for maze exploration experiments.

Model	d	Params
RNN	100	12,305
Plastic RNN	73	12,343
LSTM	46	12,379
Plastic LSTM (No Mod.)	42	12,396
Plastic LSTM (Simple Mod.)	42	12,522
HyperLSTM	30	12,251
STPN	56	12,269
FW	100	12,305
CTM	66	12,348
Plastic CTM (Ours)	49	12,207

The model architecture consists of the recurrent network followed by two separate linear heads: an actor head that outputs logits over the 4 discrete actions, and a critic head that outputs a scalar value estimate. Both heads are initialized with orthogonal weight matrices and zero biases. Actions are sampled from a categorical distribution parameterized by the actor logits.

F.4 Training with Proximal Policy Optimization

We train all models using PPO (Schulman et al., 2017) with GAE (Schulman et al., 2015) and Pop-Art normalization (Van Hasselt et al., 2016). We use 512 parallel environments to collect experience, with each environment running for 200 timesteps per episode. At the end of each training iteration, we compute returns and advantages using GAE with discount factor $\gamma = 0.99$ and trace decay parameter $\lambda = 0.95$.

The collected trajectories are then used to update the model parameters through 4 PPO epochs. During each PPO epoch, the entire batch of 512 trajectories is processed simultaneously. We use truncated backpropagation-through-time, splitting each 200-step episode into subtrajectories of length 20 and backpropagating within each, to enable the recurrent networks to learn long-term dependencies and credit assignment. The policy loss uses the PPO clipped surrogate objective with clip range $\epsilon = 0.25$, and the total loss combines the policy loss, the value loss weighted by coefficient 0.1, and an entropy bonus weighted by coefficient 0.01 to encourage exploration. Additionally, we employ KL-based early stopping (Dossa et al., 2021), where within each training iteration, we terminate the PPO epochs early if the approximate KL divergence between the new and old policies exceeds 0.03. This was disabled for STPN to allow for learning as the initial KL divergence was high.

All models are trained for 10,000 epochs using the AdamW optimizer with learning rate 3×10^{-4}. We apply gradient clipping with a maximum norm of 0.5 to stabilize training. For each model architecture, we train 3 independent runs with different random seeds and report the mean performance with standard error across seeds.

G Experimental Details: Sequential Split-MNIST

G.1 Task Configuration and Architecture

We evaluate continual learning using a modified version of the Split-MNIST task from Irie et al. (2025). The MNIST dataset LeCun et al. (1998) is split into Set A (digits 0 to 4) and Set B (digits 5 to 9). Both sets are mapped to labels 0 to 4, creating a domain-incremental setting where the model must learn distinct input-to-label mappings for each domain. At each episode, the class-to-label mapping is randomly shuffled independently for Set A and Set B.

Each episode consists of 25 images presented sequentially: 5 Set A exemplars (one per class) with labels provided, followed by 5 Set A queries (immediate learning test) with labels withheld, then 5 Set B exemplars (one per class) with labels provided, followed by 5 Set A queries (backward transfer test) with labels withheld, and finally 5 Set B queries (forward transfer test) with labels withheld. During training phases, the model receives image-label pairs. During query phases, a special "unknown label" token is provided and the model must predict the correct label based on associations formed in-context. Exemplars and queries are sampled from separate data splits to prevent leakage.

All images are processed through a shared 4-layer convolutional encoder before being fed to the recurrent networks. This encoder follows the same ProtoNet architecture (Snell et al., 2017) used in Section D.1. The encoder output is projected to dimension $d_{\text{input}} = 256$ and concatenated with a one-hot encoding of the label (or the unknown label token). This combined representation is fed to the recurrent network, which outputs logits for 5-way classification.

The loss is the sum of three cross-entropy terms computed at the query positions (Irie et al., 2025):

$$\mathcal{L} = \mathcal{L}_{\text{immediate}} + \mathcal{L}_{\text{forward}} + \mathcal{L}_{\text{backward}}, \tag{14}$$

where each term averages over the 5 queries in its respective phase. This training objective aims to ensure that the model can do immediate learning on Set A, then learn Set B without forgetting Set A.

G.2 Model Configurations and Hyperparameters

All models are parameter-matched to approximately 1.38M parameters by adjusting their hidden dimensions d. Table 15 shows the specific configurations used for each architecture. All models use $d_{\text{input}} = 256$ with identical encoders.

Table 15: Model configurations for Sequential Split-MNIST experiments.

Model	d	Params	Dropout	Learning Rate
Plastic LSTM (No Mod.)	396	1,380,978	0.2	1×10^{-4}
STPN	516	1,379,609	0.2	1×10^{-4}
Plastic CTM (Ours)	512	1,379,526	0.2	1×10^{-4}

G.3 Training and Evaluation

All models were trained for 300 epochs with a batch size of 256 using the AdamW optimizer. We applied gradient clipping with a maximum norm of 10.

We report accuracy separately for each query phase: immediate accuracy measures performance on Set A queries after seeing Set A exemplars, forward accuracy measures performance on Set B queries after seeing both Set A and Set B exemplars, and backward accuracy measures performance on Set A queries after seeing both Set A and Set B exemplars. The backward transfer metric directly measures whether the model suffers from catastrophic forgetting—a model that forgets Set A after learning Set B will achieve low backward accuracy despite high immediate accuracy.

H Computational Overhead

Table 16 reports forward pass time, forward-plus-backward pass time, and per-step auxiliary state memory for all architectures, as measured on an NVIDIA H100 GPU. Adding synchronization-based plasticity to the CTM increases forward-pass time by approximately 24% and roughly doubles the full training step, with an $\mathcal{O}(d^2)$ state memory cost from maintaining the synchronization matrix S^t. This overhead is shared across plastic architectures that maintain dense pairwise state, including the Plastic RN, FW, and STPN.

Table 16: Computational overhead of plastic and non-plastic RNNs on an NVIDIA H100 GPU. Forward pass time (Fwd), forward-plus-backward time (Fwd+Bwd), and per-step auxiliary state memory (State) are reported. The $\mathcal{O}(d^2)$ state memory of the Plastic CTM is comparable to other plastic architectures such as the STPN, FW, and Plastic RNN.

Model	Fwd (ms)	Fwd+Bwd (ms)	State (KB)
RNN	0.09	0.41	1.8
LSTM (Hochreiter & Schmidhuber, 1997)	0.16	0.75	1.5
Plastic RNN (Miconi et al., 2018)	0.25	0.48	479.9
Plastic LSTM (Simple Mod.) (Miconi et al., 2020)	0.31	0.96	16.0
Plastic LSTM (No Mod.) (Miconi et al., 2020)	0.26	1.02	121.0
HyperLSTM (Ha et al., 2016)	0.66	2.33	1.4
STPN (Rodriguez et al., 2022)	0.27	0.83	437.5
FW (Ba et al., 2016a)	0.34	0.68	842.8
CTM (Darlow et al., 2025)	0.50	1.69	18.6
Plastic CTM (Ours)	0.62	3.27	779.0

I Additional Experiment: CIFAR-10 Image Classification

To verify that the Plastic CTM retains the supervised learning capabilities of the original CTM, we evaluate it on CIFAR-10 image classification following the protocol of Darlow et al. (2025). The model processes the static image over 10 internal ticks. At each tick t, a query $q_t = Q(S^t)$ is projected from the model's synchronization and used to cross-attend to patch features produced by a convolutional backbone, yielding the input to the model o_t. The model is trained with cross-entropy loss computed at the final tick. Results are shown in Table 17. The Plastic CTM outperforms the CTM in this setting, demonstrating that synchronization-based plasticity provides benefits beyond the meta-learning regime.

Table 17: CIFAR-10 test accuracy.

Model	Test Acc.
CTM (Darlow et al., 2025)	0.848±0.001
Plastic CTM (Ours)	**0.854±0.001**