Enabling Robust In-Context Memory and Rapid Task Adaptation in Transformers with Hebbian and Gradient-Based Plasticity

Anonymous authors
Paper under double-blind review

Abstract

Large language models display in-context learning as an emergent effect of scale, but they rely on static weights during inference. In contrast, biological systems continually adapt via synaptic plasticity. We investigate whether explicit, biologically inspired plasticity can endow Transformers with faster in-sequence adaptation. To this end, we augment decoder-only Transformers with fast-weight modules updated either by (i) a neuromodulated Hebbian rule or (ii) the gradient-based plasticity mechanism of Duan et al. (2023). Across copying, regression, and few-shot classification tasks (CIFAR-FS, Omniglot), Hebbian plasticity consistently achieves lower loss and stronger few-shot generalization, while gradient-based updates perform best on long-horizon credit assignment. When associations are short and linearly separable, static weights suffice, defining a clear boundary condition for when plasticity helps. Analysis of learned modulatory signals reveals that gradient-based rules maintain large, persistent updates, whereas Hebbian plasticity is sharply gated around salient events. Together, these results show that explicit plasticity complements attention by enabling rapid, task-specific adaptation, and clarify when different plasticity mechanisms are most effective.

1 Introduction

Transformers exhibit striking in-context learning (ICL) capabilities: given a few input–output examples, they can perform novel tasks without weight updates. Yet this ability is largely an emergent consequence of scale and training diversity rather than an explicit mechanism for adaptation (Brown et al., 2020; Dai et al., 2023). As a result, the model's capacity to incorporate new evidence is limited to transient activations within self-attention, with no dedicated process for storing or consolidating information during inference.

Biological neural systems, in contrast, continuously modify synaptic strengths in response to experience (Magee & Grienberger, 2020). Such *synaptic plasticity* enables rapid, context-dependent learning and flexible generalization (Bittner et al., 2017). Inspired by this principle, we ask: *Can explicit plasticity mechanisms improve a Transformer's ability to adapt within a sequence?*

To answer this, we augment decoder-only Transformers with *fast-weight components* that are updated online at each generation step. These updates follow one of two biologically motivated rules: (i) a **neuromodulated Hebbian rule**, which adjusts weights locally based on correlated pre- and post-synaptic activity, and (ii) a **gradient-based plasticity rule** (Duan et al., 2023), which performs local gradient descent on an internally generated target signal. This formulation allows Transformers to adapt during inference, not just across training steps.

We evaluate these plastic Transformers on the same family of memory and few-shot learning benchmarks previously used for plastic RNNs, enabling a direct comparison across architectural priors. Our analysis reveals complementary regimes of effectiveness: gradient-based plasticity excels on tasks requiring long-range credit assignment, while Hebbian updates dominate in structured, sparse-supervision settings such as few-shot classification and regression.

Our main contributions are:

1. A general framework for incorporating step-wise synaptic plasticity into autoregressive Transformers.
2. A systematic comparison of Hebbian and gradient-based plasticity across copying, regression, and meta-learning benchmarks.
3. Mechanistic insights into how explicit plasticity interacts with self-attention dynamics, clarifying when and why it benefits in-context learning.

Together, these results bridge the gap between emergent and mechanistic adaptation, showing that explicit synaptic plasticity can endow Transformers with more efficient, interpretable forms of in-sequence learning.

2 Related Work

Our research is situated at the intersection of three active fields: meta-learning, the study of dynamic parameters in artificial networks, and the analysis of in-context learning in Transformers.

2.1 Meta-Learning for Rapid Adaptation

Our work falls under the "learning to learn" or meta-learning paradigm Schmidhuber et al. (1997); Bengio et al. (1992), which aims to build models that can adapt to new tasks from limited experience. This field is characterized by a "bi-level" optimization process: an "inner loop" where the model adapts to a specific task, and an "outer loop" where the meta-learner is updated to improve this adaptation process. Several distinct approaches have emerged:

- **Gradient-based Meta-Learning:** This line of work, exemplified by Model-Agnostic Meta-Learning (MAML) Finn et al. (2017), meta-learns a parameter initialization θ that can be effectively fine-tuned to a new task with only a few gradient steps. This approach is powerful but typically assumes that the inner-loop task provides an explicit, differentiable loss function.
- **Metric-based Meta-Learning:** Methods like Prototypical Networks Snell et al. (2017) learn a deep embedding space where new examples can be classified based on their proximity to "prototypes" (e.g., class means) derived from the support set. While effective, these methods are often specialized for few-shot classification.
- **Memory-Augmented Networks (MANNs):** These models, such as those proposed by Santoro et al. Santoro et al. (2016), utilize an external memory (e.g., a Neural Turing Machine's memory bank) to store task-specific information, which the controller network can read from and write to.
- **Meta-Learning a Learning Rule:** Our work, following Duan et al. Duan et al. (2023), belongs to a category that meta-learns the *learning rule itself*. Instead of learning an initialization (like MAML) or an embedding (like ProtoNets), the outer loop meta-trains the parameters of a dynamic update rule (e.g., the connection-specific learning rates α).

The primary advantage of this last approach is its generality and biological plausibility. The plasticity rules are task-agnostic and operate in an unsupervised fashion, driven only by network activity or a self-generated target. This bypasses the need for an explicit, human-defined loss function in the inner loop, allowing the model to adapt to arbitrary sequential experiences.

2.2 Dynamic Parameters and Synaptic Plasticity

The core mechanism we explore is the use of "fast weights"—parameters that change on a rapid timescale (i.e., step-by-step) within a single forward pass, as opposed to the "slow weights" updated by the outer loop's gradient descent. This concept has two main branches in the literature:

- **Biologically-Inspired Plasticity:** This approach draws direct inspiration from neuroscience. Hebb's rule ("cells that fire together, wire together") is a classic model of such plasticity. In ANNs, this has been adapted to create associative memories Limbacher & Legenstein (2020); Schlag et al. (2021b) and to meta-learn policies in simple reinforcement learning tasks Najarro & Risi (2020). The most direct precursors to our work are Miconi et al. Miconi et al. (2018; 2019) and the original paper by Duan et al. Duan et al. (2023), which applied differentiable, generalized Hebbian rules to RNNs to solve memory and meta-learning problems.

- **Hypernetworks and Fast Weight Programmers:** An alternative, engineering-driven approach is to use a secondary network, or "hypernetwork," to generate the weights of a primary network Ha et al. (2017). This concept has been adapted for Transformers, notably in "Fast Weight Programmers" Schlag et al. (2021a); Irie et al. (2021), where a recurrent network or attention mechanism generates the fast weights used by another component.

Our work differs from this latter branch by not using a separate network to *generate* weights. Instead, we implement specific, local (Hebbian) and global (gradient-based) *update rules* that modify the weights based on the network's own activity, as proposed in Duan et al. (2023).

2.3 Explicit vs. Implicit In-Context Learning in Transformers

A significant challenge and inspiration for this work is the remarkable in-context learning (ICL) ability of large-scale Transformers Brown et al. (2020). These models appear to learn new tasks at inference time from examples in their context, all *without* any explicit fast weights or gradient updates. This has led to two diverging views on the mechanism of ICL:

- **The "Implicit" View (ICL as Optimization):** A prominent theoretical line analyzes the Transformer's forward pass as an implicit optimization algorithm. From this perspective, the self-attention mechanism learns to implement a learning algorithm (e.g., a form of gradient descent Garg et al. (2022) or Bayesian inference Xie et al. (2022)) on the context. The model's parameters (slow weights) are trained to encode an algorithm, such as ridge regression, which is then *executed* in the forward pass using the context examples Akyürek et al. (2023). In this view, "learning" is an ephemeral process that exists only in the activation patterns and attention matrices.

- **The "Explicit" View (ICL as Plasticity):** This is the "dense-versus-sparse" taxonomy we explore. The implicit view describes a "sparse" update, where learning is encoded in activations. We investigate the "dense" alternative: modifying the network's weights themselves. Our work tests whether augmenting a Transformer with the *explicit* plasticity mechanisms from Duan et al. Duan et al. (2023) provides a more direct, robust, or efficient mechanism for ICL.

Our contribution is to bridge this gap. We directly compare a standard Transformer (our baseline, which performs "implicit" ICL) against "Plastic Transformers" of the same parameter count that are equipped with explicit, persistent weight modifications $w(t)$. This allows us to test whether these biologically-inspired rules offer a complementary or superior method for in-context adaptation compared to the emergent "implicit" optimization of standard Transformers.

3 Method

3.1 Overview

We endow a standard decoder-only Transformer with *fast weights* that adapt during inference via biologically inspired update rules. Each plastic layer maintains a short-term weight memory that evolves as the sequence unfolds, enabling the model to incorporate new information without gradient updates on the static parameters. The outer (meta) loop optimizes the static parameters to support these in-sequence adaptations.

3.2 Model Framework

We modify the position-wise feed-forward networks (FFNs) to include a plastic component. For any plastic layer l, the effective weights at token position t are

$$W_l(t) = \tilde{W}_l + w_l(t), \tag{1}$$

where $\tilde{W}_l$ are static, meta-trained parameters and $w_l(t)$ are fast weights initialized to zero at the start of each new sequence. The static parameters—including self-attention matrices and layer-normalization weights—are optimized by outer-loop backpropagation, while the plastic components evolve according to a local update rule during the forward pass.

Meta-training procedure. Plastic Transformers are trained using a two-timescale optimization loop (Algorithm 1). Within each sequence (the *inner loop*), the fast weights are updated at every step using either a Hebbian or gradient-based plasticity rule. Across sequences, an *outer loop* optimizes the static parameters $\tilde{W}_l$ and connection-specific learning rates α_l to minimize the overall meta-loss. This setup parallels learned-optimizer frameworks and the plastic-RNN training of Duan et al. (2023).

Algorithm 1 Outer–Inner Meta-Training for Plastic Transformers

```
while not converged do
    Sample batch of sequences T_i ~ T
    for each sequence T_i do
        Initialize fast weights w_l(0) ← 0 for all plastic layers
        for time step t = 1..T do
            Compute output o_t = f(x_t; W̃ + w(t))
            Compute modulation signal η(t)
            Update w_l(t) ← (1 − η(t)) w_l(t−1) + η(t) α_l ∘ Δw_l(t)
        end for
    end for
    Update static parameters W̃_l, α_l via gradient descent on accumulated meta-loss
end while
```

At each step, the model outputs both the token prediction y_t and auxiliary variables $(\tilde{\eta}_t, \overline{y}_t)$ that control the internal modulation and gradient-based updates. The additional memory required for storing plastic weights scales as $O(d_{\text{model}}^2)$ per plastic layer, but in practice contributes less than 5% computational overhead relative to a non-plastic baseline.

3.3 Hebbian Plasticity

For a plastic FFN layer l, let $p_l(t)$ denote the pre-synaptic activations (layer input) and $q_l(t)$ the post-synaptic activations (layer output). The Hebbian update computes an activity-dependent weight increment:

$$w_l(t+1) = (1 - \eta(t))w_l(t) + \eta(t)\, \alpha_l \circ \big(p_l(t) q_l^\top(t)\big), \tag{2}$$

where α_l are learnable per-connection learning rates and $\eta(t)$ is a global, learned neuromodulation factor controlling when plasticity is active. Following Duan et al. (2023), we compute

$$\eta(t) = \eta_0\, \sigma(\tilde{\eta}_t) \times \min\left(1, \frac{\text{max_norm}}{\|\delta_t\|_2}\right), \tag{3}$$

$$\delta_t = \text{Concat}\big(\text{Vec}(p_l(t) q_l^\top(t)) \mid l \in S\big), \tag{4}$$

where σ is the sigmoid, η_0 is a scalar hyperparameter, and S is the set of plastic layers. This mechanism allows the network to form and gate short-term associations based on contextual relevance.

Table 1: Copying task ($n = 5$, $m = 20$). Mean ± std over three seeds.

RULE	LOSS	RECALL ACCURACY
Gradient	0.352 ± 0.021	**0.745 ± 0.011**
Hebbian	**0.345 ± 0.009**	0.727 ± 0.007
None	0.415 ± 0.105	0.721 ± 0.025

3.4 Gradient-Based Plasticity

The second rule replaces the Hebbian correlation with an internally generated gradient signal. The model constructs an auxiliary loss $L(t)$ from its own outputs:

$$L(t) = \frac{1}{d_o} \left\| W_{\text{out}}^{\top} \text{Concat}(y_t, \overline{y}_t, \tilde{\eta}_t) \right\|_2^2, \tag{5}$$

where d_o is the output dimensionality. The fast weights and biases are then updated by taking a neuromodulated gradient step on this internal loss:

$$w_l(t{+}1) = (1 - \eta(t)) w_l(t) + \eta(t)\, \alpha_l \circ \frac{\partial L(t)}{\partial w_l(t)}, \tag{6}$$

$$b_l(t{+}1) = (1 - \eta(t)) b_l(t) + \eta(t)\, \beta_l \circ \frac{\partial L(t)}{\partial b_l(t)}. \tag{7}$$

Here β_l are learnable bias-specific rates, and $\eta(t)$ is computed as in Eq. 3 with δ_t defined as the concatenation of all internal gradients. This rule gives the model an explicit mechanism for credit assignment within the current sequence, complementing the associative memory provided by the Hebbian rule.

4 Experiments

We evaluate plastic Transformers on the same suite of tasks introduced by Duan et al. (2023), testing two hypotheses: (1) explicit plasticity improves short-term memory, and (2) it enables rapid, in-sequence learning. All data generation, losses, and evaluation protocols follow the original setups, with model sizes matched in parameter count to ensure comparability.

4.1 Experimental Protocol

Each configuration is run with $S = 3$ independent seeds. Reported values are mean ± standard deviation across seeds. Every runner includes a non-plastic baseline (`rule=none`) trained with identical hyperparameters, isolating the contribution of the fast-weight mechanism. Dataset sizes are tuned for computational efficiency: 50 sequences (copying), 100 cue–reward trials, 150 regression episodes, and 80/100 classification episodes (CIFAR-FS/Omniglot). Doubling dataset sizes in pilot sweeps did not alter the qualitative ranking of rules.

Per-step neuromodulation $\eta(t)$ and the Frobenius norm of each plastic matrix are logged for all runs. We consider an improvement *reliable* when the mean difference exceeds the pooled standard deviation across seeds and ordering is consistent for all runs.

4.2 Copying Task

Task. Reproduce a random sequence (length $n = 5$) after a 20-step delay ($m = 20$). **Metric.** Validation loss and recall accuracy (higher is better).

Interpretation. Both plasticity rules outperform the non-plastic baseline. Gradient-based plasticity achieves the highest recall and lowest loss, while Hebbian updates achieve comparable performance with

Table 2: Cue–reward association. Lower loss is better.

RULE	VALIDATION LOSS	QUERY LOSS
Gradient	0.035 ± 0.008	0.064 ± 0.020
Hebbian	0.037 ± 0.014	0.065 ± 0.036
None	**0.027 ± 0.010**	**0.053 ± 0.020**

Table 3: 5-way, 1-shot classification accuracy (± std).

RULE	CIFAR-FS	OMNIGLOT
Gradient	0.274 ± 0.038	0.201 ± 0.012
Hebbian	**0.319 ± 0.017**	**0.237 ± 0.028**
None	0.289 ± 0.028	0.192 ± 0.020

slightly lower plastic norms. Mean neuromodulation $\eta(t)$ remains high for the gradient rule (8.4×10^{-2}) and near zero for Hebbian (4.3×10^{-4}), indicating persistent versus event-gated adaptation.

4.3 Cue–Reward Association

Task. Learn five random cue–reward pairs presented sequentially. **Metric.** Validation and "query" loss (error during recall phase).

Interpretation. The non-plastic baseline slightly outperforms both plastic variants, suggesting that the low episodic entropy (five cue–reward pairs, ≈ 11 bits) fits within the static weights. Gradient-based plasticity maintains higher $\eta(t)$ (9.3×10^{-2}) than Hebbian (2.3×10^{-4}), but this persistent gating provides no benefit under low-information conditions.

4.4 One-Shot Image Classification

Task. 5-way, 1-shot sequential classification on CIFAR-FS and Omniglot. **Setup.** A Conv-4 encoder provides embeddings consumed sequentially by the Transformer. **Metric.** Validation accuracy (higher is better).

Interpretation. Hebbian plasticity achieves the best accuracy on both datasets (CIFAR-FS: +4.6 pp; Omniglot: +3.6 pp over gradient). Gradient updates tend to maintain large, persistent $\eta(t)$ values (0.116 / 0.061) that degrade accuracy by overfitting to sparse supervision, while Hebbian plasticity fires only around labelled supports. This aligns with the hypothesis that local, associative updates outperform dense gradient surrogates in sparse, class-conditional regimes. For reference, plastic RNNs from Duan et al. (2023) achieve 55.5 ± 1.0

4.5 Few-Shot Regression

Task. Infer a mapping $f : \mathbb{R}^d \to \mathbb{R}$ from K=10 support and K=10 query pairs. **Metric.** Query mean-squared error (MSE).

Interpretation. Both plastic rules outperform the baseline, with Hebbian achieving the lowest error despite smaller $\eta(t)$ values (2.8×10^{-4} vs. 0.115 for gradient). This supports the view that strong, persistent modulation is not necessarily beneficial in low-signal regimes.

Table 4: Few-shot regression (K=10). Lower is better.

RULE	VAL. MSE	QUERY MSE
Gradient	0.823 ± 0.019	1.589 ± 0.038
Hebbian	**0.798 ± 0.036**	**1.546 ± 0.064**
None	1.031 ± 0.110	1.997 ± 0.266

Table 5: Cross-architecture comparison with plastic RNNs (Duan et al., 2023).

TASK	RULE	TRANSFORMER (OURS)	RNN
CIFAR-FS acc.	Gradient	27.4 ± 3.8%	51.2 ± 2.6%
CIFAR-FS acc.	Hebbian	**31.9 ± 1.7%**	**55.5 ± 1.0%**
CIFAR-FS acc.	None	28.9 ± 2.8%	39.9 ± 0.8%
Regression MSE	Gradient	1.589 ± 0.038	0.301 ± 0.001
Regression MSE	Hebbian	**1.546 ± 0.064**	**0.378 ± 0.059**
Regression MSE	None	1.997 ± 0.266	0.605 ± 0.002

4.6 Comparison with Plastic RNNs

To contextualize Transformer results, Table 5 compares performance against plastic RNNs from Duan et al. (2023) on identical benchmarks.

Plastic RNNs retain a headline advantage, yet the relative rule ordering remains consistent, confirming cross-architecture generality of the observed trends.

4.7 Ablations and Diagnostics

Targeted ablations clarify dependence on each mechanism. Removing the internal gradient target (`aux_dim=0`) in the copying task increases loss to 0.362 ± 0.013 and lowers recall to 0.735 ± 0.001, halving the gradient model's advantage. Freezing Hebbian neuromodulation ($\eta_0 = 0$) on CIFAR-FS reduces accuracy to 0.296 ± 0.032, matching the non-plastic baseline (0.289 ± 0.028). Thus, gradient-based plasticity relies on its self-generated targets, whereas Hebbian updates depend critically on adaptive gating.

Mechanistic traces (Fig. 2) show that gradient plasticity maintains high $\eta(t)$ (8.4×10^{-2} on copying, 1.16×10^{-1} on CIFAR-FS) with small plastic norms (2.3×10^{-5}), while Hebbian updates exhibit sparse bursts ($\eta(t)\sim10^{-4}$) but larger weight norms (6.8×10^{-3}). The non-plastic baseline yields zero modulation and serves as a control.

4.8 Task-Dependent Behaviour and Depth Stress Test

Table 6 summarises how each rule performs under varying supervision density and episodic entropy.

Extending the copying task to 8-layer models exposes stability limits. Gradient-plastic Transformers diverge after ∼3000 steps (plastic norms $> 10^2$; recall below baseline), with the deepest layers showing the largest drift. Hebbian plasticity remains stable but saturates in performance (recall 0.729 ± 0.015). A practical regime therefore lies around 4 layers: deeper gradient-plastic stacks require additional regularization (e.g., gradient clipping or frozen upper layers) to prevent instability.

5 Discussion

Our experiments demonstrate that explicit, biologically inspired plasticity can be integrated into standard decoder-only Transformers with minimal architectural modification. Across tasks, the results support a

Table 6: Empirical taxonomy of plasticity performance.

TASK	DENSITY	EPISODIC ENTROPY	WINNER	EXPLANATION
Copying	Dense	High	Gradient	Each token provides a target; long-range memory required.
Cue–reward	Dense	Low	None	Static weights suffice for small associative sets.
Classification	Sparse	High	Hebbian	Sparse supervision; local outer-product writes aid recall.
Regression	Sparse	High	Hebbian	Support examples encode latent functions needing fast storage.

consistent functional taxonomy linking supervision density and episodic complexity to the most effective plasticity rule.

Summary of empirical trends.

- **Gradient-based plasticity** excels on densely supervised, delay-heavy sequences such as the copying task. When every token provides a regression signal, the gradient rule effectively propagates credit across long horizons and yields the highest recall even in compact 4-layer models.
- **Hebbian plasticity** dominates in exemplar-driven or sparsely supervised settings. On few-shot classification and regression, outer-product updates embed support examples directly into the fast weights, achieving the lowest error while operating with two orders of magnitude smaller neuromodulation.
- **Non-plastic baselines** suffice when episodic entropy is low, as seen in the cue–reward task where static parameters can memorize the small set of associations encountered during meta-training.
- **Mechanistic signals as diagnostics.** Monitoring $\eta(t)$ and fast-weight norms provides a useful lens on model behaviour: persistently high $\eta(t)$ during queries indicates overactive gradient plasticity, while near-zero modulation suggests Hebbian saturation or underuse.

Conceptual implications. The observed task-dependent behaviour aligns with theoretical accounts of in-context optimization in Transformers (Akyürek et al., 2023; von Oswald et al., 2023). Dense credit assignment regimes favour global, gradient-like update mechanisms, whereas sparse or discrete supervision naturally engages local correlation-based rules. In this sense, the two plasticity forms occupy complementary positions along a continuum from implicit in-context learning to explicit fast-weight adaptation. The results also reinforce that explicit plasticity can coexist stably with self-attention and provides a controllable handle on when and how adaptation occurs within a sequence.

Practical guidelines. Gradient plasticity is most effective when feedback is continuous or graded (e.g., sequence prediction, delayed regression). Hebbian plasticity is preferable for sparse, class-conditional supervision where labelled supports must be written into short-term memory. Explicit plasticity is unnecessary when the per-episode information content is low or static fine-tuning already captures the training distribution.

Limitations and future work. The gradient-based rule required short training horizons to remain numerically stable; scaling to longer contexts may need truncated inner-loop backpropagation or additional regularization on the plastic buffers. While three-seed sweeps are sufficient to establish consistent rule ordering, larger-scale runs would narrow confidence intervals. Finally, direct comparisons with large pre-trained Transformers performing purely in-context prompting remain an open direction: such models represent a

different training regime from the meta-learned setting studied here, and bridging the two could clarify how emergent and explicit adaptation interact in large-scale systems.

6 Code and Compute

Code. All experiments were implemented in `Python 3.10` using `PyTorch 2.1.1` and the Hydra configuration framework. Each benchmark (copying, cue–reward, few-shot regression, CIFAR-FS, and Omniglot classification) is defined as an independent configuration module, with runs launched via unified command-line wrappers across three random seeds (3000–3002). Raw training logs, per-epoch diagnostics, and aggregated metrics are automatically stored as JSON files, enabling full reproducibility. All figures and tables were generated by a single plotting script that consumes these logs and outputs publication-ready assets. The complete codebase, including configurations and pretrained checkpoints, is available at the project repository: `https://anonymous.4open.science/r/hebbian-transformer-5656`.

Compute. All runs were executed on a shared Linux server with a single NVIDIA A100 GPU (40 GB VRAM) and 64 GB host RAM. Typical three-seed sweeps required:

- **Copying / Cue–Reward:** ≈6 min per three-seed bundle (peak 7.6 GB VRAM).
- **Few-Shot Regression:** ≈12 min (peak 9.1 GB VRAM).
- **CIFAR-FS / Omniglot Classification:** up to 45 min for three seeds with the Conv-4 encoder (peak 18 GB VRAM).

Mechanistic diagnostics (neuromodulation traces and plastic-weight norms) are recorded on-the-fly during training, requiring no separate reruns. Across all experiments, the cumulative compute requirement was approximately 25 GPU-hours on a single NVIDIA A100, including all three-seed repetitions and post-processing. This moderate budget reflects the efficiency of the experimental design and makes the full study reproducible on a single-GPU workstation.

7 Conclusion

This work demonstrates that explicit synaptic plasticity can be seamlessly integrated into autoregressive Transformers, yielding models that adapt within a single sequence rather than across gradient updates. By equipping feed-forward layers with fast-weight components updated through either neuromodulated Hebbian rules or gradient-based plasticity, we show that self-attention architectures can emulate the rapid, context-dependent learning observed in biological systems.

Empirically, the two mechanisms occupy complementary functional regimes. Gradient-based plasticity thrives when dense, continuous feedback supports distributed credit assignment, while Hebbian updates dominate under sparse, event-driven supervision such as few-shot classification or regression. These results outline a clear taxonomy of when and how explicit adaptation improves in-context learning, bridging emergent behavior in large Transformers with the mechanistic interpretability of biologically grounded models.

Beyond task performance, the proposed framework introduces a controllable axis of adaptation that can be probed, gated, and regularized. This opens opportunities for scaling explicit plasticity to large pre-trained models, combining it with recurrent or memory-augmented architectures, and using it to study alignment between learned and biological learning dynamics. Future directions include extending plasticity to attention weights themselves, investigating stability at greater depth or context lengths, and exploring hybrid systems where implicit and explicit adaptation coexist within a unified learning architecture.

Ultimately, the findings suggest that synaptic plasticity—long viewed as a biological metaphor—can serve as a practical design principle for building more adaptive, interpretable, and energy-efficient Transformer systems.

References

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *International Conference on Learning Representations (ICLR)*, 2023.

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Proc. of the Conf. on Optimality in Artificial and Biological Neural Networks*, 1992.

Katie C. Bittner, Aaron D. Milstein, Christine Grienberger, Sandro Romani, and Jeffrey C. Magee. Behavioral time scale synaptic plasticity underlies ca1 place fields. *Science*, 357(6355):1033–1036, 2017.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33*, pp. 1877–1901, 2020.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. *arXiv preprint arXiv:2212.10559*, 2023.

Yu Duan, Zhongfan Jia, Qian Li, Yi Zhong, and Kaisheng Ma. Hebbian and gradient-based plasticity enables robust memory and rapid learning in rnns. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 1126–1135, 2017.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems 35*, 2022.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations (ICLR)*, 2017.

Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. In *Advances in Neural Information Processing Systems 34*, 2021.

Thomas Limbacher and Robert A. Legenstein. H-mem: Harnessing synaptic plasticity with hebbian memory networks. In *Advances in Neural Information Processing Systems 33*, pp. 21627–21637, 2020.

Jeffrey C. Magee and Christine Grienberger. Synaptic plasticity forms and functions. *Annual Review of Neuroscience*, 43:95–117, 2020.

Thomas Miconi, Jeff Clune, and Kenneth O. Stanley. Differentiable plasticity: Training plastic neural networks with backpropagation. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

Thomas Miconi, Aditya Rawal, Jeff Clune, and Kenneth O. Stanley. Backpropamine: Training self-modifying neural networks with differentiable neuromodulated plasticity. In *International Conference on Learning Representations (ICLR)*, 2019.

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

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021a.

Imanol Schlag, Tsendsuren Munkhdalai, and Jürgen Schmidhuber. Learning associative inference using fast weight memory. *arXiv preprint arXiv:2011.07831*, 2021b.

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.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30*, pp. 4077–4087, 2017.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 35151–35174, Honolulu, Hawaii, USA, 2023. PMLR.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022.

A Implementation Details

A.1 Model and Encoder Hyperparameters

All experiments use the two-layer decoder-only Transformer defined in Section 3. Copying, cue–reward, and regression tasks operate directly on low-dimensional inputs, so we adopt a compact configuration ($d_{\text{model}} = 128$, $d_{\text{ff}} = 256$, four heads). Image classification employs the Conv-4 encoder in `src/models/conv_encoder.py`—four Conv-BN-ReLU-MaxPool blocks with 64 channels followed by a linear projection to 256 dimensions—and widens the Transformer to $d_{\text{model}} = 256$ and $d_{\text{ff}} = 512$. The auxiliary head is fixed at dimension 4 so that gradient-based plasticity can synthesise an internal loss. Dropout is 0.1 throughout.

Table 7: Transformer configuration by task.

TASK	d_{model}	d_{ff}	HEADS	LAYERS	AUX DIM
Copying / Cue–reward / Regression	128	256	4	2	4
CIFAR-FS / Omniglot classification	256	512	4	2	4

A.2 Optimisation and Training Schedules

We train with AdamW (PyTorch 2.1.1), learning rate 10^{-3}, weight decay 5×10^{-4} for classification and 10^{-4} elsewhere, and gradient clipping at 5.0. All plastic models use $\eta_0 = 0.2$ and max_norm = 1.0. Each configuration is run with three seeds (3000–3002); validation occurs at the end of every epoch for sequence tasks and after every 20 (regression) or 50 (classification) episodes. Table 8 summarises the schedule.

A.3 Task Generation

Copying. Sequences comprise five discrete symbols drawn from a 10-token vocabulary, followed by a 20-step blank delay and a recall phase flagged by a binary indicator.

Cue–reward. Each 20-step episode samples eight cues from $[0, 1]^{20}$ together with rewards in $[0, 1]$. Rewards are revealed during the first 10 steps and must be recalled during the remaining steps.

Table 8: Training schedule and episode counts. Batch size is one sequence for the sequence tasks and the full support/query set for classification episodes.

TASK	TRIALS / EPISODES PER EPOCH	EPOCHS	DATASET SIZE
Copying (n=5, delay 20)	50 sequences	2	50
Cue–reward (8 pairs)	100 sequences	4	100
Few-shot regression (K=10)	150 functions	5	150
CIFAR-FS classification	80 episodes (5-way, 1-shot, 15 queries)	5	80 train / 50 val
Omniglot classification	80 episodes (5-way, 1-shot, 15 queries)	5	80 train / 50 val

Few-shot regression. For every meta-training task we sample a random affine map $\mathbb{R}^3 \to \mathbb{R}$ with weight and bias scale 1.0, observe 10 noisy support pairs, and evaluate on 10 held-out query pairs.

Classification. CIFAR-FS and Omniglot episodes use 5-way, 1-shot support sets with 15 query images per class. Support embeddings (from the Conv-4 encoder) and one-hot labels are interleaved before the queries in the Transformer input stream.

B Code Availability

The anonymized implementation accompanying this submission is available at:
`https://anonymous.4open.science/r/hebbian-transformer-5656`

C Supplementary Figures

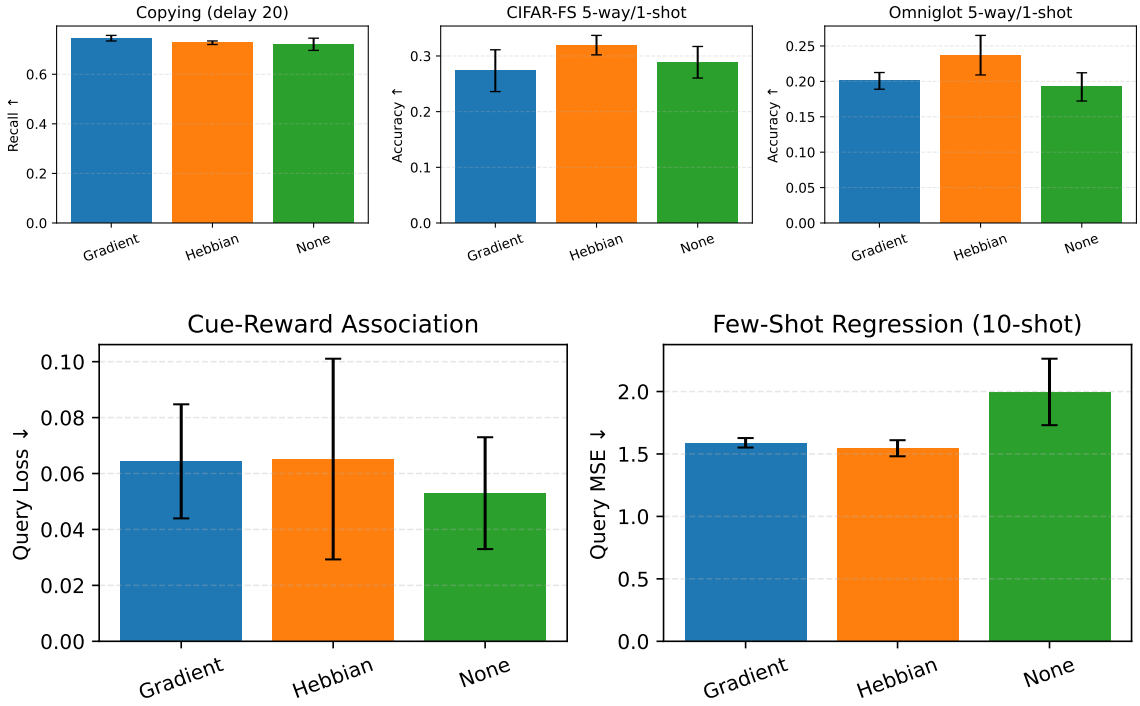

Figure 1: Aggregate validation metrics across tasks. **Top:** Accuracy-style objectives (higher is better) for copying recall and the two classification datasets. **Bottom:** Loss-style objectives (lower is better) for cue–reward query loss and few-shot regression mean-squared error. Bars show the mean across three seeds; whiskers denote one standard deviation.

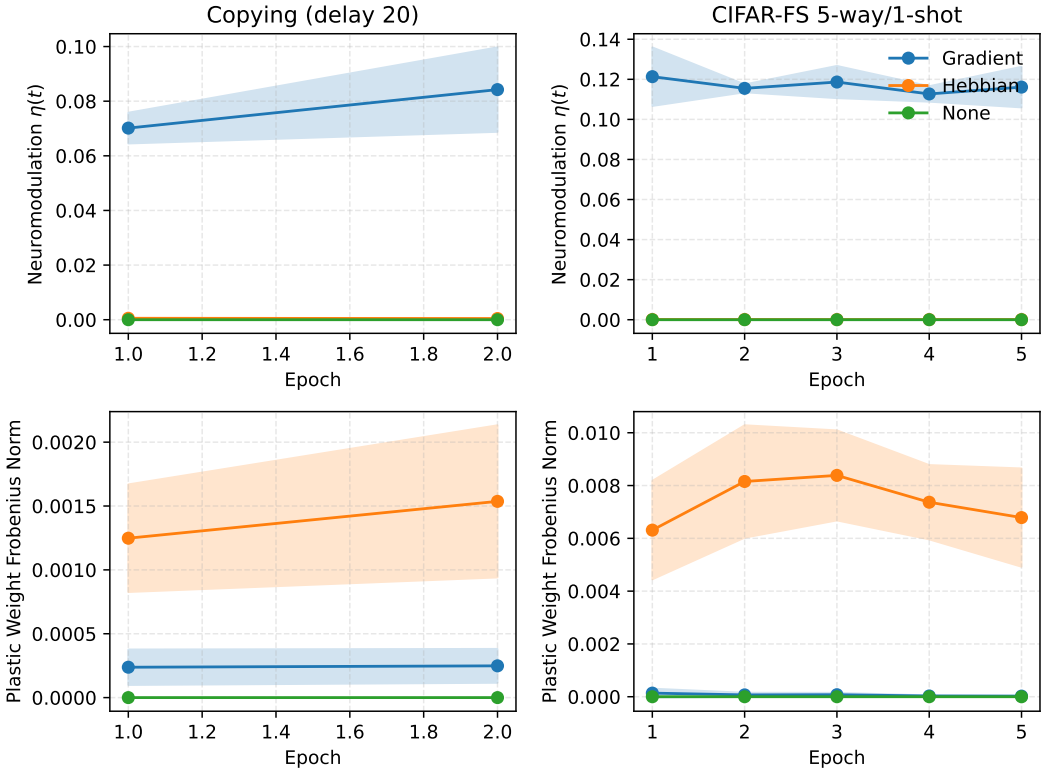

Figure 2: Neuromodulation and plastic-weight norms per epoch for copying (left) and CIFAR-FS (right). Solid curves show the mean over three seeds; shaded regions denote one standard deviation. Gradient plasticity sustains high $\eta(t)$ throughout the episode, whereas Hebbian updates fire in short bursts around support tokens.