# EPHEMERAL MEMORY: SUCCESSFUL RETENTION OF SHORT-TERM INFORMATION USING VARIABLE PLASTICITY AND DECAYING WEIGHT VALUES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

A few weights with quickly decaying values and high learning rates can replace the hidden, recurrent connection in an RNN, storing short-term memory in the parameters themselves via a standard loss calculation and backpropagation gradient calculation. In most artificial neural network (ANN) designs, the gradient update mechanism (loss function and backpropagation) is viewed as a process distinct from the forward pass, with one generating predictions and the other computing and applying parameter updates. However, error computation and backpropagation can be repurposed for use as short-term memory. This is done by designating a fraction of the model's weights as "fast weights," endowed with larger learning rates to enable swift adaptation and decay to prevent gradient instability. The result is a constantly updating record of recent gradient updates directly accessible to the network's weights, negating the need for rolling context windows or recurrent connections. This approach enables short-term context to be stored purely in ephemeral, rapidly updated parameters, offering a more direct route from manually stored memory to parameter-based memory. We demonstrate this approach on character-level sequence prediction tasks, showing that our ephemeral weights model can successfully solve key-recall and repeated sequence tasks without any recurrent connections, achieving comparable accuracy to traditional RNNs. The approach establishes that continuous state information can be propagated solely through parameter updates; hence the use of static plasticity and the removal of recurrent connections, allowing the existing weight update mechanism to encode both long and short-term information.

## 1 INTRODUCTION

While it is typical to consider the loss function and backpropagation as distinct from the forward pass components of an artificial neural net (ANN), it may be constructive to consider their permanent inclusion in the structure of the model, to be used even at inference-time. Error computation and backpropagation can be repurposed for use as a form of short-term memory, even to the point of replacing the hidden connection in a recurrent neural network (RNN). To do so necessitates varying degrees of plasticity among the parameters of the neural network, so that both long-term gradient descent can continue to improve the overall performance of the model, while faster-changing parameters grant the ability to encode short-term information. A simple way to do this is by designating a fraction of the parameters Ephemeral Weights, and giving them a larger learning rate. They also need a rate of decay in order to prevent gradient instability and eventual explosion. The result is a constantly updating record of recent gradient updates received from the loss function, directly accessible to the weights of the model. This parameter based record can replace the recurrent connection in a Recurrent Neural Network (RNN), allowing the existing weight update mechanism to encode *both* long and short-term information.

The purpose of this paper is to establish that simple ephemeral weights are able to replace a recurrent connection in sequence prediction tasks. They store short-term context in the parameters themselves of a neural network, using existing weight update algorithms. The loss function, backpropagation of errors, and gradient descent are repurposed for rapid online learning of the current task and its relevant data. The paradigm renders those weight update mechanisms inseparable from the model, even at inference time. Future neural networks likely will use a more sophisticated weight plasticity scheme, alongside the recurrent connections and context windows used in modern setups. Ephemeral weights show that there may exist a more gradual transition from the more manually stored memory to parameter-based memory.

The contributions of this paper are as follows: (1) We introduce ephemeral weights, a novel approach that repurposes standard backpropagation to store short-term context directly within a neural network's parameters, (2) We demon-

strate that designating a small fraction of network parameters as "ephemeral weights", a variant of fast weights, with higher learning rates can enable memory functions without recurrent connections, (3) We show that a controlled forgetting mechanism (forget rate) is necessary to prevent gradient instability while preserving memory capabilities, (4) We empirically demonstrate that our approach successfully outperforms key-recall and repeated sequence tasks comparable to traditional RNNs, despite lacking recurrent connections, (5) we apply this approach to a far more difficult task, palindrome completion, and analyze its performance in this context.

## 2 Related Works

### 2.1 Contextual, Online, and Meta-Learning

A variety of approaches modify neural networks to maintain information beyond their immediate inputs, either by changing parameters during inference or adding explicit memory structures. While these methods don't necessarily focus on encoding short-term memories in parameters themselves, they meet related challenges.

#### 2.1.1 Parameter-Based Adaptation

Meta-learning research focuses on creating models that can quickly adapt themselves to novel circumstances. These approaches typically involve an inner and outer training loop structure. As Finn et al. (2017) demonstrated with MAML, the outer loop optimizes for initial parameters that can rapidly adapt to new tasks in the inner loop. However, the computational challenges are significant—the outer training loop is often not well-suited for backpropagation-based gradient calculations since it would require storing activations for an infeasible number of parameters. Thus, evolutionary learning or other alternatives are commonly used, with traditional backpropagation typically limited to very small inner loop models.

In continual and online learning, models must adapt to new data without forgetting previous knowledge—a capability that standard neural networks struggle with due to the rigid training/inference paradigm. A common approach is experience replay, where past experiences are stored and periodically revisited. Popularized in deep Q-networks by Mnih et al. (2015), replay buffers stabilize training and improve data efficiency by breaking temporal correlations in online data. However, this solution adds complexity through buffer management and sampling strategies.

As recent works confirm Wang et al. (2024), memory-based rehearsal provides a strong baseline in lifelong learning but requires extra storage and training iterations that simpler online learners would avoid. More sophisticated replay variants (selective storing, compressed activation replay, or generative replay) further increase system complexity by introducing components solely to manage past knowledge. This represents a fundamental trade-off: simplicity versus the ability to retain information across learning episodes.

#### 2.1.2 External Memory Architectures

Rather than modifying learning algorithms, some approaches augment neural networks with explicit memory structures. Neural Turing Machines (NTM), introduced by Graves et al. (2014), pair a neural controller with a read-write memory matrix, allowing networks to store and retrieve information over extended sequences. This design enables algorithmic tasks like copying or sorting that standard RNNs cannot perform. However, the controller must learn where to write or read, making training more resource-intensive and complex. While subsequent variants like the Differentiable Neural Computer refined these mechanisms, they still struggled with consistency and scalability over very long sequences.

Building on these foundations, Weston et al. (2014) introduced Memory Networks for question-answering tasks, using a separate memory to store facts with an iterative attention process for retrieval. Similarly, Santoro et al. (2016) showed that Memory-Augmented Neural Networks (MANN) can rapidly encode new samples into external memory, achieving one-shot learning without extensive retraining. The trade-off is an extra memory module and controller logic that simpler feed-forward networks don't require.

In recent years, transformer-based models have also incorporated external memory systems. Wu et al. (2022) proposed Memorizing Transformers that maintain a key-value store of past activations, extending the effective context window beyond fixed self-attention limits. Similarly, Wang et al. (2023) augmented language models with dedicated long-term memory modules that require learning separate read/write operations. These approaches improve knowledge retention but add complexity through additional networks and procedures.

#### 2.1.3 Current Industry Approaches and Limitations

The mainstream AI industry often approaches these challenges backward, with a renewed focus on inference-time compute that optimizes for the exclusion of training loop components. Fine-tuning serves as a workaround for the constraints of static, non-learning models. Parameter-efficient fine-tuning techniques allow for more space-efficient incorporation of new knowledge, while retrieval-augmented generation (RAG) addresses limitations in context window capacity by retrieving external information. Attention mechanisms on limited-length context windows solve short-term memory challenges by simply preserving all information and performing expensive relevance computations.

Recent state-space models like Mamba and Hyena attempt to address these limitations but create lossy representations of context. RWKV aims to apply attention-like mechanisms to true RNNs but still faces long-term dependency challenges similar to LSTM, GRU, and vanilla RNN architectures—causing generated outputs to easily veer from context.

We can view the attention mechanism itself as a form of meta-learning that dynamically generates weights for a linear layer. In contrast, our approach with ephemeral weights generates only changes to a limited number of highly flexible parameters, potentially offering a more direct and efficient route to parameter-based short-term memory without the architectural complexity of dedicated memory modules or the algorithmic overhead of experience replay.

### 2.2 Dynamic Parameter Adaptation and Fast Weights

The concept of rapidly updated parameters in neural networks has been explored through various complementary approaches. These techniques enable online adaptation by making parameters dynamic, effectively treating short-term weight changes as a form of memory. While traditional neural networks maintain static weights after training, techniques like fast weights, hypernetworks, and differentiable plasticity allow certain parameters to update rapidly in response to new inputs or tasks, adding a form of in-network memory stored in temporary weight changes.

Fast weights have a long history in neural network research, dating back to Hinton & Plaut (1987), who described a dual weight architecture where each weight is mirrored by one with a higher learning rate that decays toward zero. The 'effective' weight for each forward pass is the sum of these fast and slow weights, with the original goal being to *deblur* old associations rather than explicitly encode short-term information. Schmidhuber (1992) further developed this concept, drawing inspiration from biological short-term synaptic plasticity to create secondary weights that update based on recent activity.

Modern implementations of this idea take various forms. Miconi et al. (2018) implemented differentiable Hebbian plasticity, where each synapse has a plastic component adjusted with a Hebb rule during forward passes, with the *degree* of plasticity learned through gradient descent. This enables the network to memorize recent patterns by temporarily modifying weights, allowing it to succeed at tasks like one-shot recall that static networks fail at. Their plastic recurrent networks demonstrated the ability to memorize high-dimensional images seen during testing, whereas non-plastic counterparts couldn't—highlighting the benefit of this added complexity.

Hypernetworks extend this idea by using one neural network to generate the weights of another. Fast Weight Programmers (FWP) apply this concept to compute weight changes at each step, with a meta network continually emitting rapid parameter updates for the main network. As Irie et al. (2021) demonstrated, these rapid updates can serve as an alternative to recurrent connections, with current enhancements making FWPs competitive with modern architectures. Ba et al. (2016) took a different approach, generating fast weights based on both a hidden recurrent connection and decaying previous fast weights, replacing the hidden vector with one calculated through an iterative inner loop using a fast memory weight matrix.

While promising, hypernetworks introduce efficiency and learning challenges through their outer/inner loop structure. Each training iteration may require maintaining two sets of weights—the hypernetwork's and the generated network's—along with their respective gradients, significantly increasing memory usage and computational cost. The need to backpropagate through both loops can limit scalability, often necessitating specialized optimization strategies or approximations.

Another approach is dynamic evaluation, introduced by Mikolov et al. (2010) and refined by Krause et al. (2018) for language models. Here, the model performs extra gradient updates on recent data during inference to improve next-word predictions. This yields lower perplexity on sequences by continuously adapting model parameters to recent tokens, at the cost of 3× or more computational overhead during inference. Dynamic evaluation makes inference much more complex (almost like a mini-training) compared to static models, since each step requires on-the-fly backpropagation.

Finn et al. (2017) formalized a meta-learning approach (MAML) to train models that can be quickly fine-tuned with a few gradient steps on a new task. While MAML doesn't add new network modules, it alters the training procedure to initialize the model in a state amenable to rapid parameter changes. This yields a form of short-term memory (the quick weight update) for new task information, though it requires second-order optimization during training and careful hyperparameter tuning.

A more recent innovation is Fast Weight Layers (FWL) by Clark et al. (2022), which added to Transformers to emulate dynamic evaluation via learned linearized gradient updates. The FWL component learns to take the model's recent activations and "update" a set of fast weights through an attention mechanism that mirrors gradient descent, effectively giving the model a built-in one-step adaptation mechanism. This improves performance on language modeling, especially for rare or repeating tokens, but introduces an additional neural module atop the base model and a two-pass processing of each input.

Interestingly, multi-head attention can itself be viewed as a special case of fast weight programming. In attention, the "fast weights// are effectively the key-value associations—an ephemeral weight matrix computed on the fly for each sequence of queries, then used immediately to transform them. This mirrors the fast-weight concept: a separate mechanism rapidly produces a parameter-like matrix to be used in the next computation step.

Overall, dynamic parameter strategies provide short-term memory by adjusting weights in response to new data, blurring the line between learning and inference. The recurring theme is that they outperform static models on adaptation-heavy tasks, but at the cost of more elaborate model architectures or training routines (e.g., meta-learning loops, auxiliary update networks) to manage these on-the-fly changes. The approach explored in this paper builds on these foundations, focusing on a minimal implementation of ephemeral weights that repurposes existing gradient calculation mechanisms for short-term memory storage.

## 3 METHODS

### 3.1 BIOLOGICAL INSPIRATION

Our approach draws inspiration from biological mechanisms of synaptic plasticity, particularly Long-Term Potentiation (LTP). In biological neural networks, LTP induces a temporary strengthening of synapses that lasts approximately 1-6 hours, operating on a timescale between immediate neural firing and permanent structural changes. This process is mediated by calcium ion buildup, which both modulates the degree of plasticity and increases the likelihood of LTP occurring. This biological system provides a middle ground between immediate neural activation and permanent structural changes, allowing for information to be stored temporarily in the connection strengths themselves.

The ephemeral weights approach mimics this biological process by implementing parameters that rapidly adjust to recent inputs but gradually return to baseline—similar to how enhanced synaptic efficacy from LTP decays over hours if not reinforced. Just as the brain employs multiple timescales of memory storage (from millisecond neural firing to hours-long LTP to permanent structural changes), our model incorporates both fast-changing and stable parameters to create a more biologically plausible memory system.

### 3.2 MODEL ARCHITECTURE AND IMPLEMENTATION

Our experimental setup compares a traditional RNN with a novel model using ephemeral weights to store short-term memory. Both networks are character-level predictors trained on synthetic datasets designed to test memory capabilities.

#### 3.2.1 BASELINE RNN MODEL

The baseline is a standard RNN with a 3-layer architecture:

$$h_t = \text{ReLU}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{1}$$
$$y_t = \text{softmax}(W_{hy}h_t + b_y) \tag{2}$$

Where $h_t$ represents the hidden state at time $t$, $x_t$ is the one-hot encoded input character, $y_t$ is the output prediction, and $W_{xh}$, $W_{hh}$, $W_{hy}$ are weight matrices for input-to-hidden, hidden-to-hidden (recurrent), and hidden-to-output connections respectively. Each hidden layer contains 256 units.

#### 3.2.2 EPHEMERAL WEIGHTS MODEL

Our proposed model eliminates the recurrent connection ($W_{hh}h_{t-1}$) entirely, replacing it with a mechanism for storing information in rapidly updating parameters. The architecture is:

$$h_t = \text{ReLU}(W_{xh}x_t + b_h) \tag{3}$$
$$y_t = \text{softmax}(W_{hy}h_t + b_y) \tag{4}$$

Unlike traditional neural networks where parameters are fixed during inference, our model continues to update parameters even during evaluation, creating a parameter-based memory system.

### 3.3 EPHEMERAL WEIGHTS MECHANISM

The core innovation is introducing variable learning rates across the network's parameters. Each weight $w_k$ is paired with a plasticity parameter $\alpha_k$ determining how quickly it responds to gradient signals. Parameters are divided into two categories:

1. **Slow weights** ($\alpha_k = 1$): Standard parameters that learn long-term patterns
2. **Ephemeral weights** ($\alpha_k = 10^4$ or $10^5$): Highly plastic parameters that rapidly encode recent information

We randomly designate approximately 10% of the total parameters as ephemeral weights, excluding the final output layer. The weight update rule follows:

$$w_k(t+1) = w_k(t) - \alpha_k \cdot \nabla_k L(t) \tag{5}$$

Where $\nabla_k L(t)$ is the gradient of the loss function with respect to weight $w_k$ at time step $t$.

### 3.4 FORGETTING RATE MECHANISM

To prevent gradient instability from the rapid weight changes, we implement a controlled decay of ephemeral weights:

$$w_k^{\text{fast}}(t+1) = \gamma \cdot w_k^{\text{fast}}(t+1) \tag{6}$$

Where $\gamma = 0.7$ is the forgetting rate coefficient applied after each standard update. This mechanism ensures that information stored in ephemeral weights gradually fades, maintaining stability while allowing the network to adapt to new inputs.

### 3.5 TRAINING PROCEDURE

Networks are trained using cross-entropy loss, with targets being the next character in the sequence. Both models use simple stochastic gradient descent (SGD) without additional optimizers. The learning rate is typically set to $1 \times 10^{-4}$ for both models, though experiments with the baseline RNN also use $1 \times 10^{-3}$ in some cases as noted in the Results section.

During both training and inference, the Ephemeral Weights model computes gradients and updates parameters after every character prediction, unlike traditional models where parameters remain fixed during inference. This continuous update cycle allows short-term contextual information to be encoded directly in the model parameters.

However, I will establish that continuous state information can be propagated solely through parameter updates; hence the use of static plasticity and the removal of recurrent connections.

## 4 EXPERIMENTS

### 4.1 KEY-RECALL

A synthetic dataset is constructed for simple, single character retrieval. The sequence contains a character, '?', indicating that the next input contains a value to be stored. Later in the sequence, another character, '!', requests the stored value. The gaps are filled with 0s. Some examples:

```
0000?10!1
00000?200!2
00?,00!,
00?.0!.
```

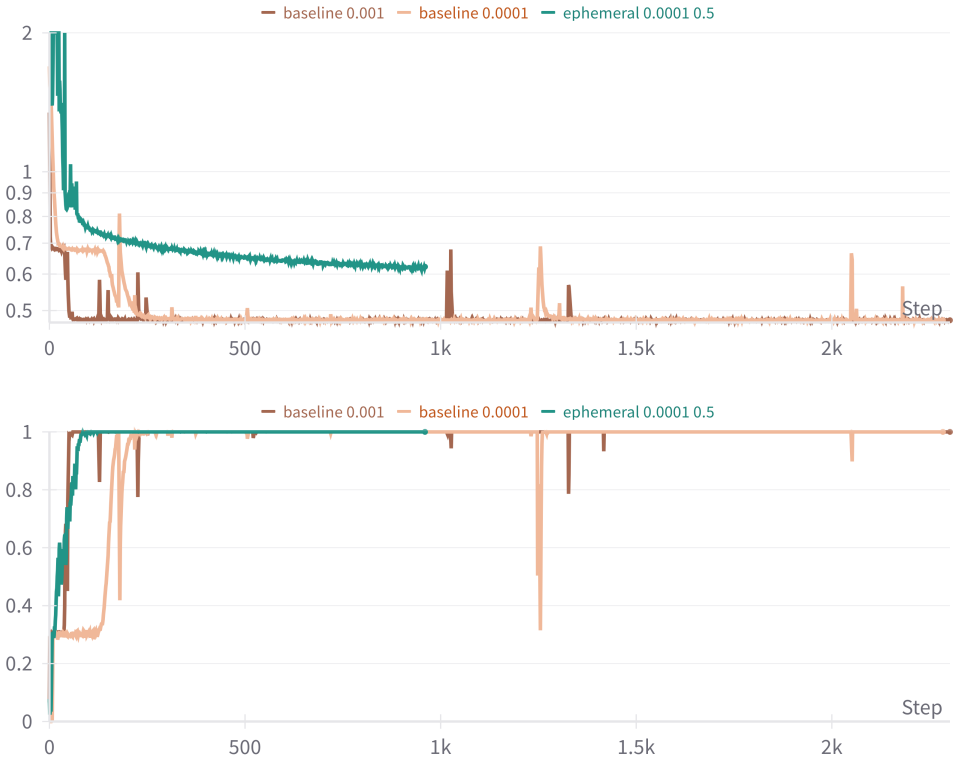


Figure 1: Baseline runs display the learning rate beside them, and ephemeral runs include the forget rate, as well. The first graph shows the loss, and the bottom graph displays the model's prediction accuracy. The baseline RNN achieves a lower loss value, and remains stable enough for a higher learning rate to be used, but Ephemeral Weights converges to a high accuracy before the baseline does, when the same learning rate is used and its forget rate is tuned well. Under experimental implementations, Ephemeral Weights does not compute as many steps in the same amount of time, but remains remarkably stable.

### 4.2 Repeated Sequences

A synthetic dataset of one to four randomly chosen characters repeated until the character limit. The ability to train this task indicates that the memory storage mechanism is capable of storing several input-output associations.

### 4.3 Palindromes

A synthetic dataset random-character palindromes. In the next-character prediction task, the model must recall the first half of characters in reverse order, despite the previous character provided in the input being only useful as an indicator of position in the reverse sequence. The network must not only store these associations but also retrieve them in inverted order. This reversal operation goes beyond a simple key-value lookup and demonstrates the network's ability to both store and transform sequential data. The ability to do so takes more than simple associative retrieval.

While the baseline model shows some capacity for this task, our Ephemeral Weights approach reveals key limitations. These challenges likely arise from several factors. First, the palindromes task inherently requires second-order temporal encoding, yet our ephemeral weights primarily capture a single state—even with added temporal embeddings. Second, the simplicity of associative memory means that, although multiple characters are stored, the system lacks

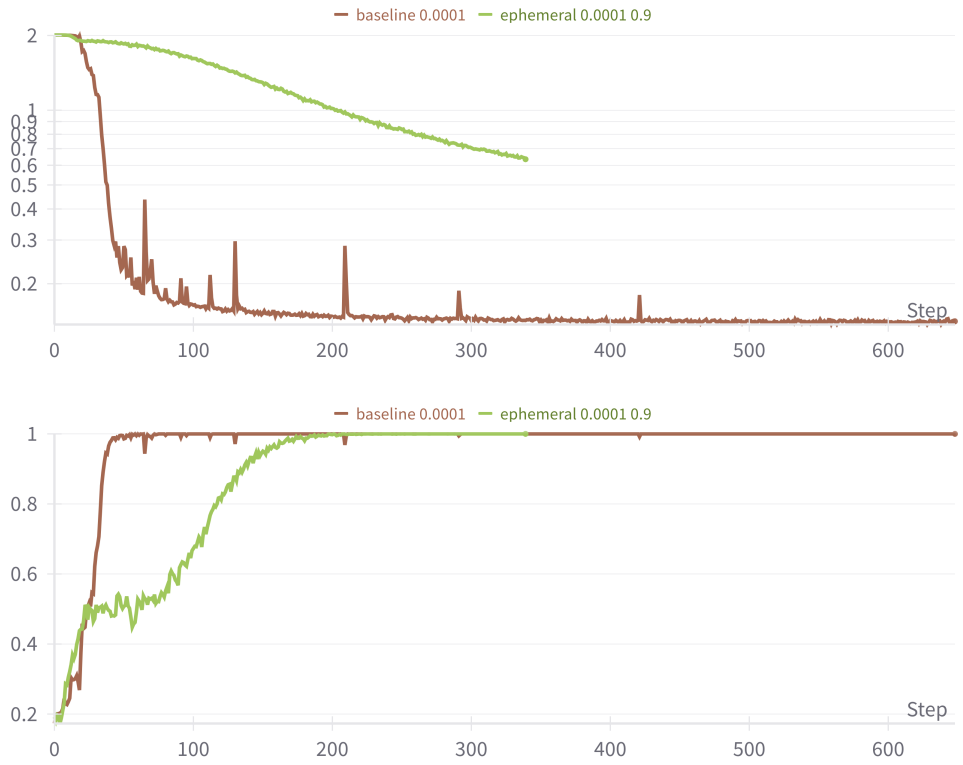


Figure 2: The baseline, traditional RNN with backprop is used as a benchmark, converging quickly on the loss values and accuracy. Ephemeral weights converge more slowly, but do so without any recurrent connection.

a built-in mechanism for transforming or reversing that order. Finally, the aggressive gradient amplification needed for rapid memory encoding appears to trigger a volatile feedback loop on the steep loss landscape of palindromes, destabilizing weight values. These interrelated issues suggest that while our method can store and retrieve sequential data, it struggles with the order-sensitive transformation required by palindromes—a point we explore further in the Discussion section.

## 5 Discussion

### 5.1 Why Palindromes Fail

The ephemeral weights model comes close to converging on a solution, but quickly becomes unstable and fails. We expect it to succeed at the task, but some refinements are needed to address the constraints of the problem. Some of these constraints appear to extend beyond this model, into limitations of associative memory as a whole.

The first possibility is that the palindromes task, by nature, is a second-order Markov process that demands explicit temporal encoding—something that our current use of associative, fast ephemeral weights does not fully capture. In a second-order Markov process, the probability of the next state depends on two preceding states:

$$P(s_t \mid s_{t-1}, s_{t-2}) \tag{7}$$

However, our model primarily leverages fast, ephemeral weights to encode a memory trace from the most recent gradient update, effectively approximating the state as:

$$P(s_t \mid s_{t-1}) \tag{8}$$

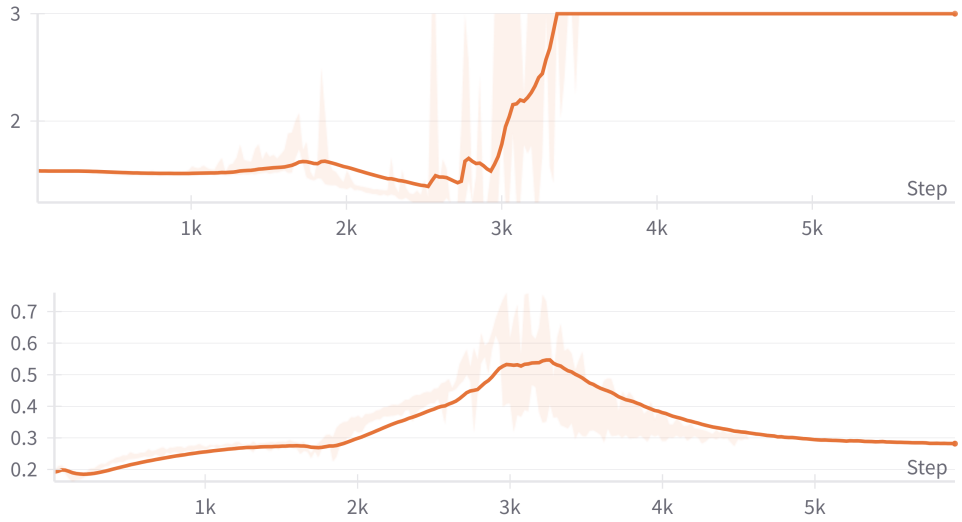


Figure 3: Baseline (not shown) converges on this problem slower than on the others. Ephemeral weights fails to remain stable enough to do so, reaching an accuracy as high as 0.75, before failure.

This formulation works well for tasks like repeated sequences or key-recall, where storing multiple associations is sufficient. Yet, in the palindromes task, knowing only $s_{t-1}$ is not enough to determine whether the sequence is in its forward or reverse phase. The correct disambiguation requires the tuple $(s_{t-2}, s_{t-1})$ to capture the necessary order information.

To address this, temporal embeddings were introduced to explicitly encode the order of inputs. Although this addition improved the model's maximum achievable accuracy by providing richer temporal context, it did not resolve the inherent stability issues. This suggests that while temporal embeddings help mitigate the underdetermination of sequence order, the fundamental challenge remains: our current state storage mechanism—despite being capable of forming key-value associations—lacks the capacity to robustly encode the nuanced temporal relationships required by a second-order process.

A second possibility is that the associative memory built into our ephemeral weights isn't naturally equipped to perform transformations—like sequence reversal—on the stored information. While these weights reliably form key-value associations (as seen in repeated sequences and key-recall), they lack an inherent mechanism to re-map or rearrange the stored content. In the palindromes task, even if multiple characters are stored, the network must output them in reverse order—a transformation that goes beyond mere recall.

The simplicity of associative memory is both its strength and its weakness. Traditional models—such as Hopfield networks—are designed for robust pattern matching and retrieval, not for algorithmically manipulating stored sequences. This aligns with observations from large LLMs. For example, when attempting a word-level reversal of a common passage (eg.. the Gettysburg Address), these models struggle significantly unless they first recite the passage in its forward order, presumably to maintain the necessary context. This behavior supports the notion that parametric memory, with its inherently associative nature, faces challenges when tasked with explicit transformations.

Thus, while our model can effectively store sequential data, its reliance on associative memory may inherently limit its ability to perform order-sensitive operations like reversal—a shortcoming shared by parametric memory in general.

The third possibility arises from the volatile feedback loop introduced by significantly amplifying gradient signals for use as an advantageous memory encoder. Synthetic datasets already have the potential to create exceptionally 'bumpy' loss surfaces, and palindromes appear to produce an exceptionally cliff-shaped loss curve. A highly effective memory encoding may lead to rapid improvements in loss values that, in turn, trigger a destabilizing explosion of ephemeral weight values, then activations, then long-term weight values. The cooling mechanism—which has been used to prevent such explosions—appears insufficient here. It may be that the cooling reduces the encoded memory to unusable levels, or that it slows convergence too much, suggesting that further experiments with scaling (e.g., longer training on slower parameters, gradient or weight clipping, or even switching to smoother tasks like natural text next-character prediction) might be required.

### 5.2 Bridging Manual Memory Storage and Parameter-Based Memory

Parameter-based memory is stored over the course of the pretraining of the model, encoding patterns, grammar rules, and facts from large text datasets, for example. Memory stored manually is stored by the model or the software built around the model, either by maintaining context windows or by recurrent connections. It is quite distinct from parameter-based memory under the current paradigm. Ephemeral weights show that there may exist a more gradual transition from the more manually stored memory to parameter-based memory. Future work should focus on plasticity values that are more dynamically selected, perhaps via a metalearning outer loop. Variable, more sophisticated learning rates may be selectable based off of only information available pre-training, such as the magnitude of high-plasticity values. Biologically-inspired learning rules beyond standard backpropagation, such as Hebbian plasticity, may stabilize volatile learning caused by ephemeral weights. A more rigorous examination of the dynamics between the forgetting rate, the ratio of ephemeral to slow weights, the plasticity of the ephemeral weights, and the base learning rate, will allow for a more principled selection of their values instead of large hyperparameter sweeps.

## 6 Conclusion

This paper demonstrates that short-term memory in neural networks can be effectively stored in rapidly adapting parameters, without relying on traditional recurrent connections. By designating a small fraction of weights as "ephemeral"—with high plasticity and controlled forgetting rates—we enable the network to encode contextual information directly in its parameters through standard backpropagation. Our approach succeeds on key-recall and repeated sequence tasks, though it reveals limitations with more complex transformations like palindrome generation.

This work blurs the lines between traditional parameter-based memory (encoded during pretraining) and manually stored memory (maintained through recurrent connections or attention mechanisms). Ephemeral weights represent a middle ground based on biological plasticity, where information storage exists on a continuum of timescales rather than a binary distinction between training and inference.

Future work should explore dynamic plasticity parameters, variable forgetting rates, and hybrid architectures combining ephemeral weights with traditional memory mechanisms, such as Hebbian learning rules. Incorporating multiple timescales of parameter adaptation may lead to more efficient, adaptable systems that better mirror the flexibility of biological memory.

The potential for this work extends to alignment and continual learning applications. A language model that is capable of continuous alignment in a more biologically plausible way has the potential to adapt more closely to individuals' needs in an ever-refining way. Smooth transitions from short to long-term memory, stored efficiently in model parameters will grant AI a new modality for computing problems over varying timescales, opening up larger and more complex projects to AI assistance. The possibilities include agents capable of coherently planning and orchestrating projects over the course of days and weeks, rather than minutes.

## References

Jimmy Ba, Geoffrey Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using Fast Weights to Attend to the Recent Past, December 2016. URL http://arxiv.org/abs/1610.06258. arXiv:1610.06258 [stat].

Kevin Clark, Kelvin Guu, Ming-Wei Chang, Panupong Pasupat, Geoffrey Hinton, and Mohammad Norouzi. Meta-learning fast weight language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 9751–9757, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.661. URL https://aclanthology.org/2022.emnlp-main.661/.

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 - volume 70*, ICML'17, pp. 1126–1135. JMLR.org, 2017. Place: Sydney, NSW, Australia Number of pages: 10.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *ArXiv*, abs/1410.5401, 2014. URL https://api.semanticscholar.org/CorpusID:15299054.

Geoffrey E. Hinton and David C. Plaut. Using Fast Weights to Deblur Old Memories. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 9(0), 1987. URL https://escholarship.org/uc/item/0570j1dp.

Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. In *Proceedings of the 35th international conference on neural information processing systems*, Nips '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 978-1-7138-4539-3. Number of pages: 15 tex.articleno: 590.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of neural sequence models. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th international conference on machine learning*, volume 80 of *Proceedings of machine learning research*, pp. 2766–2775. PMLR, July 2018. URL https://proceedings.mlr.press/v80/krause18a.html.

Thomas Miconi, Kenneth Stanley, and Jeff Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 3559–3568. PMLR, July 2018. URL https://proceedings.mlr.press/v80/miconi18a.html. ISSN: 2640-3498.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech 2010*, pp. 1045–1048. ISCA, September 2010. doi: 10.21437/Interspeech.2010-343. URL https://www.isca-archive.org/interspeech_2010/mikolov10_interspeech.html.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 1476-4687. doi: 10.1038/nature14236. URL https://www.nature.com/articles/nature14236. Publisher: Nature Publishing Group.

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*, pp. 1842–1850. PMLR, June 2016. URL https://proceedings.mlr.press/v48/santoro16.html. ISSN: 1938-7228.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, January 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.1.131. URL https://doi.org/10.1162/neco.1992.4.1.131. tex.eprint: https://direct.mit.edu/neco/article-pdf/4/1/131/812242/neco.1992.4.1.131.pdf.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In *Proceedings of the 37th international conference on neural information processing systems*, Nips '23, Red Hook, NY, USA, 2023. Curran Associates Inc. Number of pages: 14 Place: New Orleans, LA, USA tex.articleno: 3259.

Zhenyi Wang, Enneng Yang, Li Shen, and Heng Huang. A comprehensive survey of forgetting in deep learning beyond continual learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. Publisher: IEEE.

J. Weston, S. Chopra, and Antoine Bordes. Memory Networks. *CoRR*, October 2014. URL https://www.semanticscholar.org/paper/Memory-Networks-Weston-Chopra/71ae756c75ac89e2d731c9c79649562b5768ff39.

Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *The tenth international conference on learning representations, ICLR 2022, virtual event, april 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=TrjbxzRcnf-. tex.bibsource: dblp computer science bibliography, https://dblp.org tex.timestamp: Sat, 20 Aug 2022 01:15:42 +0200.