Ephemeral Memory: Successful Retention of Short-Term Information Using Variable Plasticity and Decaying Weight Values

Jaden Lorenc
Computer Science Department
Brigham Young University
Provo, USA
jaden.lorenc@byu.edu

***Abstract*—Error computation and backpropagation can be repurposed for use as short-term memory. In most artificial neural network (ANN) designs, the backward step updates only the slowly changing parametes of the network, while the forward step computes only the ephemeral activations of the network and thus providing the only viable mechanism for short-term memory. However, the backward pass can do it, too; a few weights with quickly decaying values and high learning rates can store short-term memory in the parameters themselves of an ANN via the standard loss calculation and backpropagation gradient calculation. Drawing on the biological principles of synaptic facilitation and long-term potentiation, we designate a fraction of the model's weights as "ephemeral weights," endowed with larger learning rates and decay, to enable swift adaptation and to prevent instability. The result is a constantly updating record of recent neural updates that are directly accessible to the network, negating the need for rolling context windows or recurrent connections. We demonstrate this approach functioning on progressively complex character-level sequence prediction tasks without explicitly passing any information forward. We establish that continuous state information can be propagated solely through parameter updates, allowing the existing weight update mechanism to encode short-term information where previously, only long-term information could be stored. Just as basic biological neural networks are capable of retaining information on various timescales, groundwork is laid for ANNs to also have the variety of memory mechanisms needed for continuous lifelong learning.**

***Index Terms*—neural networks, memory mechanisms, synaptic plasticity, recurrent networks, parameter adaptation, fast weights**

I. Introduction

While it is typical to consider the loss function and backpropagation as distinct from the forward pass components of an artificial neural net, 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 connection between sequential forward passes of most modern AI algorithms, such as the hidden connection present in the simple recurrent neural network (RNN) or the context window of a Transformer at inference time. To do so necessitates varying degrees of plasticity among the parameters of the neural network, so that 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.

The purpose of this paper is to establish that there exists a more gradual transition from manually stored memory to parameter-based memory; future neural networks likely will use a more sophisticated weight plasticity scheme alongside the recurrent connections and context windows used in modern setups.

The contributions of this paper are:

- The ephemeral weights mechanism,
- Ephemeral weights can enable memory functions without recurrent connections on several tasks,
- Mechanistic insight into the volatility dynamics of ephemeral weight networks,
- Dual-timescale learning can be achieved, weights of differing plasticity performing in tandem,
- Comprehensive analysis of implementation requirements including gradient propagation methods, batching constraints, and stabilization strategies essential for practical experimentation.

II. Related Works

Existing techiques encode memory either in activations (context windows, recurrent connections, external memory modules) or in parameters (fine-tuning, hypernetworks, fast weights).

A. Activation-Based Memory Mechanisms

While we focus on techniques that unify the training and inference phases of AI development, the mainstream AI industry often approaches these challenges from the opposite direction, excluding training loop components to preserve inference-time compute. Attention mechanisms preserve all information

and perform expensive relevance computations. Retrieval-augmented generation addresses limitations in context window capacity by retrieving external information, but fails to fully incorporate it. Recent state-space models like Mamba attempt to address these limitations but create lossy representations of context [1]. 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 [2].

B. Parameter-Based Adaptation

In modern 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. Fine-tuning serves as a workaround for the constraints of static, non-learning models, but continual usage of it leads to catastrophic forgetting. A common approach is experience replay, where past experiences are stored and periodically revisited. Popularized in deep Q-networks by [3], 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, without fundamentally changing the risks of catastrophic forgetting.

As recent works confirm [4], 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. Ethereal weights are an approach toward a simpler, more unified memory system.

1) External Memory Architectures: Some approaches augment neural networks with explicit memory structures. Neural Turing Machines pair a neural controller with a read-write memory matrix, allowing networks to store and retrieve information over extended sequences [5]. Subsequent variants like the Differentiable Neural Computer refined these mechanisms but both still struggle with consistency and scalability over very long sequences [6]. Building on these foundations, other methods access their external memory through iterative attention processes or metalearned read/write operations, even achieving one-shot learning without extensive retraining [7], [8]. More recently, Memorizing Transformers maintain a key-value store of past activations, extending the effective context window beyond fixed self-attention limits [9]. Other long-term memory modules are, themselves, neural nets, and require learning separate read/write operations [10]. While all such external memory architectures indeed improve knowledge retention, they do not fully reconcile the difference in methods for information storage across different timescales in the way that ethereal weights does, and scalability remains a concern.

2) Dynamic Parameter Adaptation and Fast Weights: 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 [11], 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. [12] 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. [13] 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. The network memorizes recent patterns by constantly updating 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, showing the benefit of this added complexity, but still unable to influence its own updates.

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 [14] demonstrated, these rapid updates can serve as an alternative to recurrent connections, with current enhancements making FWPs competitive with modern architectures. [15] 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 like these introduce efficiency and learning challenges through their outer/inner loop structure. Each training iteration may require maintaining two sets of weights, both 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. Ephemeral weights do not add the additional complexity that FWPs do, encoding both fast and slow weights in the same backward pass.

Another approach is dynamic evaluation, introduced by [16] and refined by [17] 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.

[18] created 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 does create a model with an optimal initialization for fast adaptation. Under our paradigm of gradients-as-memory, this approach would be similar to ours in principle, at least during its exceptionally shortened finetuning phase, though it requires second-order optimization during training and careful hyperparameter tuning. Evolutionary learning or other alternatives are commonly used, with traditional backpropagation typically limited to very small inner loop models.

A more recent innovation is Fast Weight Layers (FWL) by [19], 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 is like fast weights: a separate mechanism rapidly produces a parameter-like matrix to be used in the next computation step. However, 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 other approaches.

III. Methods

A. 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 2-3 hours in the first phase and from several hours to weeks in the second phase, 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 [20]. This biological system provides a middle ground between immediate neural activation and permanent structural changes, allowing for itinerant information to be stored temporarily in the connection strengths themselves [21].

Faster forms of synaptic plasticity are synaptic facilitation and augmentation, occurring on the order of hundreds of milliseconds to tens of minutes, all involving transient calcium dynamics in the presynaptic terminal [22].

The ephemeral weights approach mimics these biological processes by implementing parameters that rapidly adjust to recent inputs but gradually return to baseline in a manner similar to how synaptic potentiation/augmentation decays 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.

B. 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. We choose a simple, 3-layer RNN because state-of-the-art neural networks most commonly use a form of recurrent connections in order to store relevant short-term information. Diffusion models immediately encode the output into a partially denoised image, and Transformers immediately encode the output into a token that is appended to the original input, and both major models pass those partially finished works to the next step. Modifications to the simple, traditional RNN lays the groundwork for the addition of a new, in-weights modality for short-term information encoding, that may be applicable to a diverse array of more modern techniques.

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.

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.

C. 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 α_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. For ephemeral weights using Direct Feedback Alignment (DFA) [23], $\nabla_k L(t)$ represents the feedback signal computed via fixed random projection matrices rather than the true backpropagated gradient. If we were to use backpropagation, the high-plasticity weights (which have large values) would be used to backpropagate gradients, introducing significant volatility into the learning dynamics. DFA's fixed random feedback weights provide a stable gradient signal regardless of the volatile state of the high-plasticity parameters.

D. 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.

While reminiscent of the LSTM's forget gate, this forgetting rate is static and uniform across all high-plasticity ephemeral weights. Additionally, it operates on the weights themselves of the network rather than a hidden cell state value. Selective retention measures are foregone in favor of our more unified information storage mechanism.

For the purposes of stable experimentation, the high-plasticity weights are initialized to 0 at each new sequence.

E. 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 1×10^{-4} for both models, though experiments with the baseline RNN also use 1×10^{-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.

For short-term information to be effectively encoded, a complete training step—including loss calculation and backward pass—must be performed after each character prediction. This character-by-character update cycle is essential for the memory encoding mechanism, as it allows recent inputs to be immediately incorporated into the high-plasticity weights before the next prediction.

Traditional batching approaches fail with ethereal weights because they cause working memory to be shared across different sequence instances, obfuscating the information that should be encoded in high-plasticity weights. Instead, batching must be implemented using separate weight instances for each sequence, with gradient accumulation occurring only after complete sequences are processed. This ensures that each sequence maintains its own isolated working memory state.

IV. Experiments

A. 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!.
```

Successful completion of this task indicates that the model can store and retrieve a single value, not necessarily preceded by the input that originally preceded it, indicating a working memory function that isn't purely associative alone. This is an important test for a weights based memory mechanism, which would otherwise be expected to blindly encode recent inputs without the ability to selectively retrieve them later. It also tests the model's ability to balance the dual demands on its optimization on the loss function: to minimize loss while also creating a strong enough gradient signal to encode the selected memory. Under experimental implementations, Ephemeral Weights does not compute as many steps in the same amount of time, but remains remarkably stable. Its performance on this task is shown in Fig. 2, demonstrating its higher floor for loss values that's likely tied to the strength of its memory encoding. Despite the higher loss, it achieves full accuracy more quickly than the baseline RNN at the same learning rate. A deeper exploration of performance advantages, however, were deemed out of scope for this initial exploration of the mechanism, because our primary goal is to establish the viability of the approach as a memory mechanism, not as an optimizer.

B. Palindromes

A synthetic dataset of random-character palindromes tests the model's ability to perform more complex memory operations. In this next-character prediction task, the model must recall the first half of characters in reverse order to complete the palindrome. Some examples:

```
abcba
12321
qwerttrewq
```

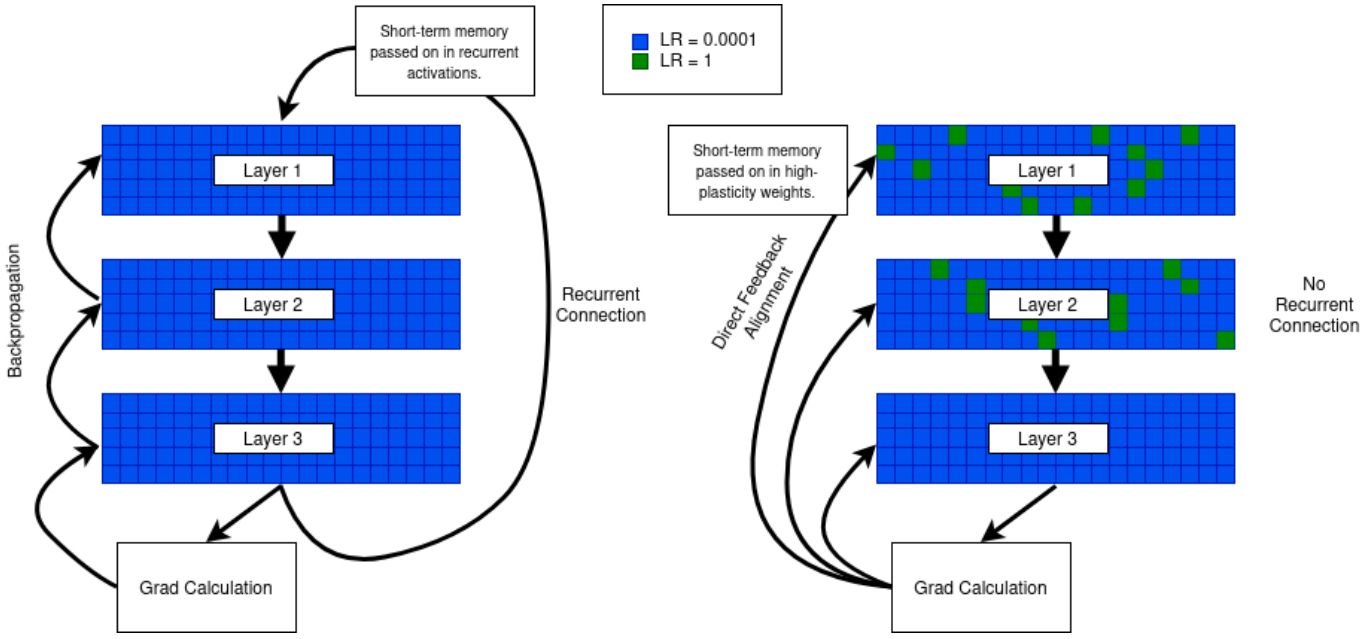

Fig. 1. In a traditional RNN, the hidden state h_{t-1} carries information forward through time. In the Ephemeral Weights model, this recurrent connection is removed, and short-term memory is instead stored in rapidly updating parameters (ephemeral weights) that adjust based on recent inputs. This allows the model to retain context without explicit recurrence.

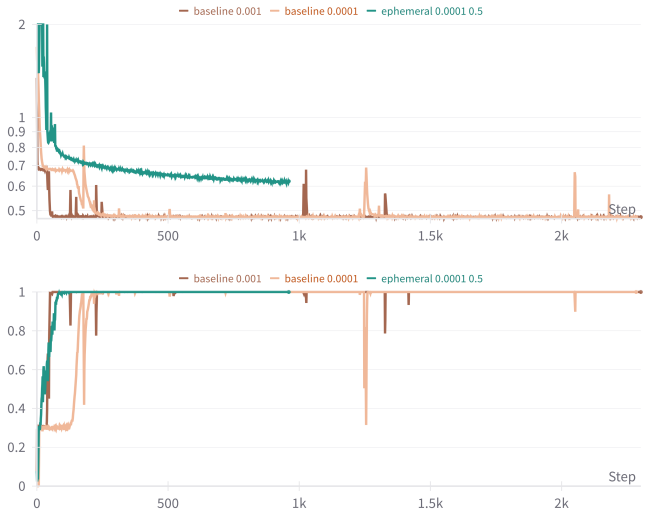

Fig. 2. Performance on the Key-Recall task. Baseline runs display the learning rate beside them, and ephemeral runs include the forget rate, as well. The first graph shows the loss, and the bottom graph displays the model's prediction accuracy. The baseline RNN achieves a lower loss value, and remains 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. Both runs end at the same time limit.

This task entails more than simple associative retrieval, requiring the network to not only store character associations but also retrieve them in inverted order. The reversal operation demands both memory storage and transformation capabilities, making it significantly more challenging than key-recall or repeated sequences.

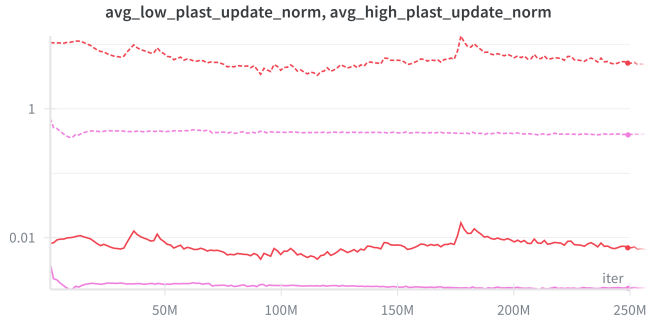

Fig. 3. Gradient norm evolution throughout training shows clear separation between high-plasticity and low-plasticity weight categories. The high-plasticity weights (red) exhibit characteristic volatility with occasional spikes corresponding to memory encoding events, while low-plasticity weights (blue) maintain steady, controlled gradient magnitudes. This separation demonstrates that the two learning processes operate independently, allowing stable long-term learning to proceed alongside volatile working memory encoding.

C. Preservation of Long-Term Learning

A critical concern with the ethereal weights approach is whether the volatile behavior of high-plasticity weights interferes with the stable, long-term learning that should occur in the low-plasticity parameters. Our analysis demonstrates that these two learning processes can coexist effectively, with each operating at its designated timescale without mutual interference.

Fig. 10 reveals that gradient norms for high-plasticity and low-plasticity weights remain highly distinct throughout training, indicating that the two categories of parameters operate in separate dynamic regimes. The high-plasticity weights ex-

hibit the expected high rate of change needed for memory encoding events, while the low-plasticity weights maintain steady, controlled gradient magnitudes characteristic of stable optimization.

This separation is crucial for the dual-timescale learning hypothesis underlying ethereal weights. The steady decrease in loss observed across our experiments suggests that the model continues to learn long-term patterns at the low-plasticity learning rate while simultaneously managing working memory at the high-plasticity rate. This indicates that ethereal weights successfully implement a form of multi-timescale learning where different parameter subsets serve distinct functional roles. The preservation of long-term learning can be attributed to only a small fraction (typically 10-20%) of parameters dedicated to volatile memory functions, as well as the forgetting mechanism preventing high-plasticity weights from accumulating permanent biases that could interfere with the network's overall optimization trajectory.

D. Volatility Emergence and Gradient Explosion

The ethereal weights mechanism exhibits characteristic volatility patterns that closely resemble exploding gradient phenomena. In both classical exploding gradients and our ethereal-weights volatility, instability stems from positive feedback in the update rule: as parameter magnitudes increase, the induced gradients scale up, amplifying subsequent updates unless explicit damping (e.g., weight decay, normalization, or clipping) keeps the effective gain below unity.

The model may achieve promising accuracy before explosion, which manifests as a cascading failure where gradient norms and weight magnitudes accelerate simultaneously, leading to rapid increases in loss and corresponding drops in accuracy. This volatility pattern is particularly pronounced with synthetic datasets, which tend to create non-smooth, "bumpy" loss landscapes that amplify the destabilizing effects of rapid weight changes.

In stable training runs, the ratio of gradient norms between high-plasticity and low-plasticity weights gradually shrinks over time, as the slow-training weights provide a stabilizing influence. However, in models that eventually explode, this gradient ratio consistently grows from the start, indicating that high-plasticity weights are gaining disproportionate influence over the network's dynamics.

Traditional stabilization techniques, however, are ineffective in this context. Gradient clipping, despite its effectiveness in traditional neural networks, fails because the memory mechanism fundamentally depends on occasional large gradient magnitudes to encode information effectively. Similarly, batch normalization and layer normalization actively harm performance by normalizing away the extreme bimodal gradient distribution that enables memory encoding.

There is a corridor of effective parameter values that maintains both the encoding power and stability of the system. A Hyperparameter sweep on the 3-character reversed sequence task indicated that the linear interaction of the base learning rate and the high-plasticity learning rate determines both the performance and the stability outcomes. It also indicated that of the two hyperparameters, high-plasticity learning rate is far more dominant in determining the accuracy of the model in the time-constrained training regime. Therefore, a balance must be struck by setting the high-plasticity learning rate to be high enough to enable effective memory encoding, but low enough to avoid instability (Fig. 7).

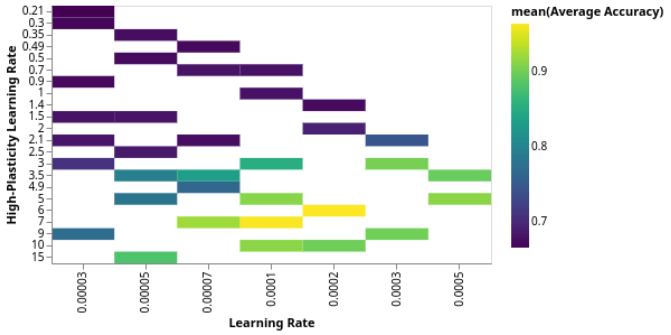

Fig. 4. Hyperparameter sweep results on the 3-character reversed sequence task showing the interaction between base learning rate and high-plasticity learning rate. The visualization demonstrates the linear relationship between these hyperparameters in determining stability outcomes, with the high-plasticity learning rate showing dominant influence over model accuracy in time-constrained training regimes.

V. Discussion

Large Language Models have surpassed human performance on a variety of tasks, yet still fail on many tasks routinely done by even a household cat. Such failure modes include long-term coherence in an agentic task, quick incorporation of new information, management of memory over long runs, and robustness to adversarial attacks such as camoflage, all hinging on the same shortcoming: the lack of a functional progression of state. A biological system, such as that of a simple mammal, has countless mechanisms for storing memory, persisting state, and managing plasticty that current LLMs only crudely approximate with its only two available systems: parameter-based memory and activation-based memory. Rather than exclusively relying on the former for long-term knowledge and the latter for short-term context, ethereal weights present a functional fluid memory system.

A. Ethereal Weights as Functional Working Memory

Our experiments demonstrate that ethereal weights successfully encode functional working memory without explicit gate-keeping mechanisms. Unlike traditional memory architectures that employ gates, attention mechanisms, or explicit memory curation, ethereal weights operate through pure associative storage of recent gradient updates. This approach can reliably hold multiple pieces of information simultaneously with positional encoding providing additional organizational structure. The memory mechanism operates as an associative system that performs best with strong retrieval cues, though the temporal decay ("age") of stored information can also serve as an effective retrieval mechanism, possibly allowing the model to make use of recency effects.

The model's volatility characteristics are linked to this memory function. The gradient explosion patterns that threaten stability are not merely implementation artifacts but fundamental consequences of the memory encoding mechanism. The aggressive gradient amplification required for rapid memory formation creates a delicate balance between effective encoding and system stability, suggesting that practical implementation requires managing that volatility. This relationship also potentially limits how low the loss can decrease while maintaining effective working memory capabilities.

B. Toward a Multi-Scale Memory Pipeline

The current ethereal weights approach represents only one component of what could become a comprehensive memory hierarchy. The limitations observed in our experiments, particularly the lack of information selection, suggests that ethereal weights are best understood as part of a larger memory pipeline rather than a complete solution.

We envision a biologically-inspired memory architecture that operates across multiple timescales: context windows for immediate information access, recurrent connections for short-term rehearsal loops, high-plasticity weights for rapid encoding, medium-plasticity weights for intermediate storage, and low-plasticity weights for long-term consolidation. This hierarchy mirrors human memory processes, where information flows from immediate perception through various stages of consolidation. Consider the analogy of remembering a phone number: humans typically have an immediate context window of approximately four digits, requiring active rehearsal (analogous to recurrent loops) to maintain longer sequences. With repetition, the number can be retained for minutes (high-plasticity storage), and with further practice, it may persist for hours or days (medium-plasticity) before potentially becoming permanently encoded (low-plasticity). Each stage serves a specific function in the overall memory system.

This multi-scale approach could enable emergent memory consolidation behaviors similar to replay buffers in reinforcement learning or bootstrapped fine-tuning in language models. A conversational AI system might initially store user preferences in context windows, transfer frequently mentioned preferences to recurrent loops, gradually encode them in high-plasticity weights, and eventually consolidate important patterns into the model's long-term parameters. This would create a natural progression from ephemeral working memory to permanent knowledge, freeing up short-term memory resources for new information.

C. Future Work

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.

More rigorous testing will also be necessary to verify effectiveness against catastrophic forgetting. By measuring performance on both the primary task and working memory tasks throughout training, researchers could quantify how effectively the two learning processes coexist. Additionally, ablation studies could examine performance when high-plasticity weights are periodically frozen or reset, allowing direct measurement of the low-plasticity weights' learning progress in isolation. Training ethereal weights models on tasks with known long-term dependencies (such as copying tasks with very long delays), while simultaneously requiring short-term memory operations, may provide stronger evidence that the architecture can maintain stable long-term learning while providing functional working memory capabilities. Preliminary results excluded from this paper suggest that sinusoidal plasticity values, akin to transformers' positional encodings but changing across time, may enhance the model's ability to balance these dual objectives.

VI. 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, establishing that parameter-based working memory can function as an effective alternative to traditional recurrent architectures while revealing important insights about the dynamics and implementation requirements of such systems.

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

[1] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces." [Online]. Available: http://arxiv.org/abs/2312.00752

[2] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, L. Derczynski, X. Du, M. Grella, K. Gv, X. He, H. Hou, P. Kazienko, J. Kocon, J. Kong, B. Koptyra, H. Lau, J. Lin, K. S. I. Mantri, F. Mom, A. Saito, G. Song, X. Tang, J. Wind, S. Woźniak, Z. Zhang, Q. Zhou, J. Zhu, and R.-J. Zhu, "RWKV: Reinventing RNNs for the transformer era," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 14 048–14 077. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.936/

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[4] Z. Wang, E. Yang, L. Shen, and H. Huang, "A comprehensive survey of forgetting in deep learning beyond continual learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[5] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *ArXiv*, vol. abs/1410.5401, 2014.

[6] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, Oct. 2016. [Online]. Available: https://doi.org/10.1038/nature20101

[7] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-Learning with Memory-Augmented Neural Networks," in *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, Jun. 2016, pp. 1842–1850.

[8] J. Weston, S. Chopra, and A. Bordes, "Memory Networks," *CoRR*, Oct. 2014.

[9] Y. Wu, M. N. Rabe, D. Hutchins, and C. Szegedy, "Memorizing transformers," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[10] W. Wang, L. Dong, H. Cheng, X. Liu, X. Yan, J. Gao, and F. Wei, "Augmenting language models with long-term memory," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. Nips '23. Red Hook, NY, USA: Curran Associates Inc., 2023.

[11] G. E. Hinton and D. C. Plaut, "Using Fast Weights to Deblur Old Memories," *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 9, no. 0, 1987.

[12] J. Schmidhuber, "Learning to control fast-weight memories: An alternative to dynamic recurrent networks," *Neural Computation*, vol. 4, no. 1, pp. 131–139, Jan. 1992.

[13] T. Miconi, K. Stanley, and J. Clune, "Differentiable plasticity: Training plastic neural networks with backpropagation," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Jul. 2018, pp. 3559–3568.

[14] K. Irie, I. Schlag, R. Csordás, and J. Schmidhuber, "Going beyond linear transformers with recurrent fast weight programmers," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. Nips '21. Red Hook, NY, USA: Curran Associates Inc., 2021.

[15] J. Ba, G. Hinton, V. Mnih, J. Z. Leibo, and C. Ionescu, "Using Fast Weights to Attend to the Recent Past," Dec. 2016.

[16] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Interspeech 2010*. ISCA, Sep. 2010, pp. 1045–1048.

[17] B. Krause, E. Kahembwe, I. Murray, and S. Renals, "Dynamic evaluation of neural sequence models," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, Jul. 2018, pp. 2766–2775.

[18] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1126–1135.

[19] K. Clark, K. Guu, M.-W. Chang, P. Pasupat, G. Hinton, and M. Norouzi, "Meta-learning fast weight language models," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 9751–9757.

[20] M. A. Lynch, "Long-term potentiation and memory," *Physiological Reviews*, vol. 84, no. 1, pp. 87–136, Jan. 2004.

[21] S. J. Martin, P. D. Grimwood, and R. G. Morris, "Synaptic plasticity and memory: An evaluation of the hypothesis," *Annual Review of Neuroscience*, vol. 23, pp. 649–711, 2000.

[22] W. G. Regehr, "Short-Term Presynaptic Plasticity," *Cold Spring Harbor Perspectives in Biology*, vol. 4, no. 7, p. a005702, Jul. 2012.

[23] A. Nøkland, "Direct Feedback Alignment Provides Learning in Deep Neural Networks."

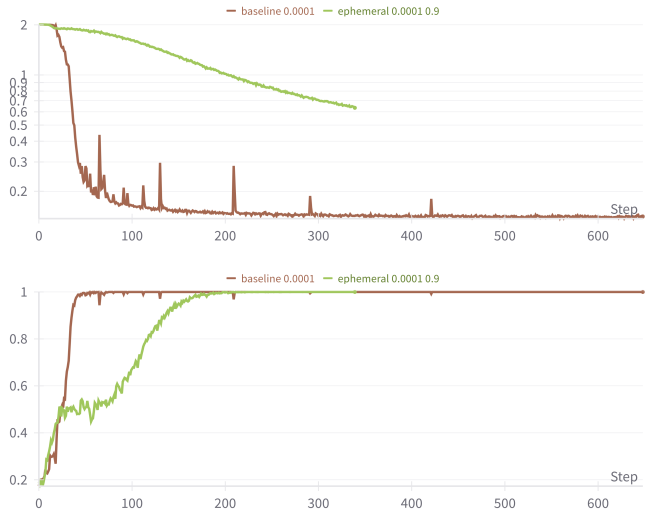

Fig. 5. Performance on the Repeated Sequences task. The baseline, traditional RNN with backprop is used as a benchmark, converging quickly on the loss values and accuracy. Ephemeral weights converge more slowly, but do so without any recurrent connection.

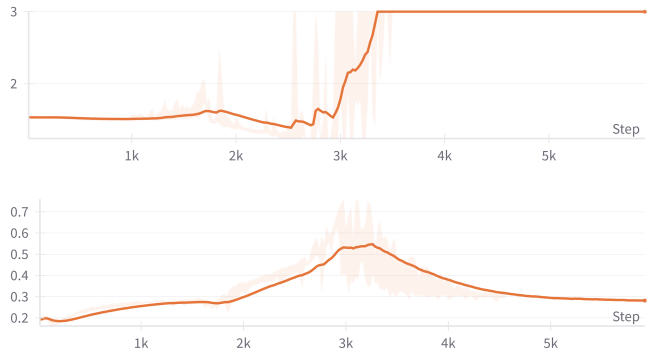

Fig. 6. Ephemeral weights demonstrate characteristic volatility on the palindromes task, reaching accuracy as high as 0.75 before experiencing gradient explosion and failure. This pattern illustrates the inherent instability that can emerge with synthetic datasets that create particularly steep loss landscapes.

VII. Appendix

A. 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. Meant to test the model's ability to store multiple association in the same parametric working memory simultaneously, as an assurance that the mechanism isn't severely limited in capacity. The mechanism is capable, yet appears to hit a saddle point in performance until later in the training, hypothesized to be due to learning how to reconcile overlapping representations in the weights, as shown in Fig. 5. Again, the difference in loss values is laid bare, indicating the model's unique relationship with gradient magnitudes. Overall, the ephemeral weights mechanism is proved capable of storing these associations without recurrent connections, the implications of which include new avenues for optimization, since the recurrent connection requires sequential operations that limit parallelism, in turn limiting scaling potential.

1) Volatility Emergence and Gradient Explosion: The palindromes task reveals a basic challenge with ethereal weights: their susceptibility to gradient explosion under certain conditions. As shown in Fig. 6, the model can achieve promising accuracy (up to 0.75) before experiencing catastrophic failure characterized by rapidly increasing loss and plummeting accuracy. This volatility pattern is particularly pronounced with synthetic datasets, which tend to create non-smooth, "bumpy" loss landscapes that amplify the destabilizing effects of rapid weight changes.

Our analysis reveals that gradient explosion in ethereal weights follows a predictable pattern driven by the ratio of gradient norms between high-plasticity and low-plasticity weights. In stable training runs, this gradient ratio tends to shrink over time as the slower weights provide stabilizing influence. However, in models that eventually explode, the gradient ratio consistently grows from the start, indicating that high-plasticity weights are gaining disproportionate influence over the network's behavior.

This observation suggests that gradient explosion in ethereal weights is not merely a case of traditional exploding gradients, but rather a feedback loop where high-plasticity weights exploit the learning objective in an uncontrolled manner. The rapid weight changes create increasingly large gradient signals, which in turn drive even more extreme weight updates, creating a runaway process that overwhelms the stabilizing influence of slower weights.

2) Hyperparameter Dependencies and Effective Learning Rate: The stability of the system depends on a delicate balance between multiple factors:

Base Learning Rate: Controls the only weights whose values truly persist across time steps. If these slow weights are too weak to provide adequate stabilizing influence over the more volatile high-plasticity weights, the system becomes unstable.

High-Plasticity Learning Rate: Creates a narrow operational window: too low and the model lacks effective working memory, too high and gradient explosion becomes inevitable.

Ratio of High to Low-Plasticity Weights: Typically set to 0.1 or 0.2, this ratio must provide sufficient capacity for associative memory storage while avoiding the instability that comes with too many volatile parameters.

Forgetting Rate: Must balance memory persistence with stability: too aggressive and ethereal memory disappears before it can be utilized, too conservative and gradient explosion becomes unavoidable.

A critical challenge is that appropriate values for these hyperparameters appear highly task-dependent, influenced by factors such as the required memory persistence duration, the density of information that must be stored in working memory, and the natural volatility of the dataset itself. This dependency

on intensive hyperparameter tuning represents a significant limitation for practical applications.

A Hyperparameter sweep on the 3-character reversed sequence task indicated that the linear interaction of the base learning rate and the high-plasticity learning rate determines the stability outcomes. It also indicated, in the same sweep, that of the two hyperparameters, high-plasticity learning rate is far more dominant in determining the accuracy of the model in the time-constrained training regime. Therefore, a balance must be struck, when choosing hyperparameters, setting the high-plasticity as high enough to enable effective memory encoding, but low enough to avoid instability (Fig. 7).

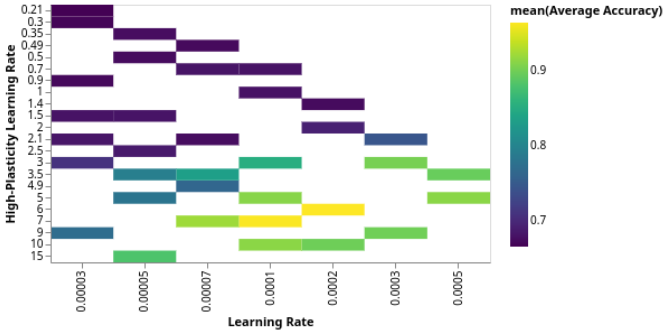

Fig. 7. Hyperparameter sweep results on the 3-character reversed sequence task showing the interaction between base learning rate and high-plasticity learning rate. The visualization demonstrates the linear relationship between these hyperparameters in determining stability outcomes, with the high-plasticity learning rate showing dominant influence over model accuracy in time-constrained training regimes.

B. Configuration Performance Summary

Following extensive experimentation with different model configurations, we present a high-level comparison of performance across various combinations of model type, updater mechanism, and recurrence settings.

The key differentiator between RNN and Ethereal implementations is the per-batch-item versus shared weights architecture. While standard RNNs maintain synchronized weights across all batch items, the Ethereal implementation maintains separate weight instances for each sequence in the batch. This architectural difference allows ethereal weights to store sequence-specific information directly in the parameters, effectively replacing the need for recurrent connections.

The stability advantage of DFA over backpropagation becomes particularly apparent in the Ethereal configurations. When using backpropagation, the high-plasticity weights (which have large values) are used to backpropagate gradients, introducing significant volatility into the learning dynamics. DFA avoids this issue by using fixed random feedback weights, providing a stable gradient signal regardless of the volatile state of the high-plasticity parameters.

C. Volatility and Stability Analysis

The ethereal weights mechanism exhibits characteristic volatility patterns that closely resemble exploding gradient phenomena. In both classical exploding gradients and our ethereal-weights volatility, instability stems from positive feedback in the update rule: as parameter magnitudes increase, the induced gradients scale up, amplifying subsequent updates unless explicit damping (e.g., weight decay, normalization, or clipping) keeps the effective gain below unity. Understanding these dynamics is crucial for practical implementation and reveals fundamental constraints of the approach.

1) Gradient Explosion Dynamics: The volatility manifests as a cascading failure where gradient norms and weight magnitudes accelerate simultaneously, leading to rapid increases in loss and corresponding drops in accuracy, as seen in Fig. 6. This behavior appears intrinsic to the mechanism rather than a mere implementation artifact—the aggressive gradient amplification required for effective memory encoding creates inherent instability.

A key early warning indicator emerges from monitoring the gradient norms of high-plasticity and low-plasticity weights. Stability correlates with this value shrinking and stabilizing over time, while growth of the gradient norm precedes explosive behavior (Fig. 8). This observation aligns with the intuitive understanding that high-plasticity weights, when left unchecked by slower stabilizing weights, can enter a positive feedback loop where rapid changes amplify subsequent gradient signals.

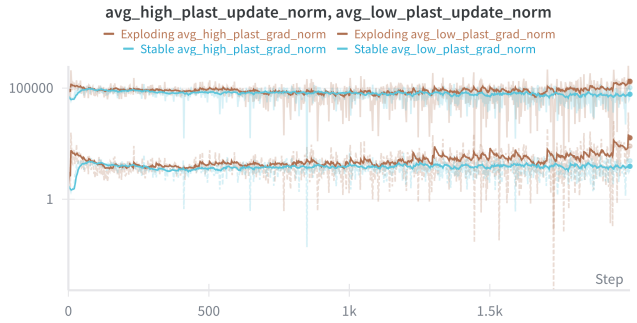

Fig. 8. A learning rate of 0.0001 is used for the stable run, and the exploding run uses a learning rate of 0.0002. The gradient norms of the exploding run (brown) grow jointly, while the stable run's (blue) gradient norms stabilize. The ratio of high-plasticity to low-plasticity gradient norms shrink, slightly, in the exploding run, but remain stable in the stable run.

Holding other hyperparameters constant, varying the base learning rate and the plasticity modifier reveals a linear boundary between stable and unstable regimes. Lower base learning rates allow for higher plasticity modifiers before instability arises, while higher base learning rates necessitate more conservative plasticity settings (Fig. 9). A limitation of this experiment is that some runs that initially appear stable within the training timeframe still exhibit climbing loss numbers at the tail end, suggesting eventual instability if training were extended further.

2) Forgetting Rate Dynamics: The forgetting rate mechanism proves essential but insufficient for complete stabilization. While aggressive forgetting rates prevent explosion,

TABLE I
PERFORMANCE COMPARISON ACROSS MODEL CONFIGURATIONS

Model	Updater	Recur.	Perf.	Key Reason	Mechanism
RNN	DFA	Any	Fail	Poor performance	DFA an imprecise approximation of Backprop
RNN	Backprop	True	Good	Standard NN training	Hidden state carries temporal info
RNN	Backprop	False	Poor	No memory mechanism	No hidden state, no weight adaptation
RNN	BPTT	True	Good	Traditional RNN with BPTT	Gradients flow through time
RNN	BPTT	False	Poor	Delayed batch gradient descent	No temporal connection to exploit
Ethereal	DFA	True	Good	Immediate weight adaptation	Fixed feedback weights stabilize volatile updates
Ethereal	DFA	False	Good	Per-batch adaptation	Each batch maintains separate weights
Ethereal	Backprop	Any	Poor	Rapid gradient explosion	High-plast weights used in backward pass
Ethereal	BPTT	True	Mixed	Rapid gradient explosion	High-plast weights used in backward pass
Ethereal	BPTT	False	Fail	No within-sequence adaptation	BPTT delays updates, negating ethereal benefit

[a]Ethereal models maintain per-batch-item weight sets, enabling memory storage without recurrence.
The success of ethereal weights with DFA stems from the stability provided by fixed random feedback weights, while backpropagation configurations work but with higher volatility.

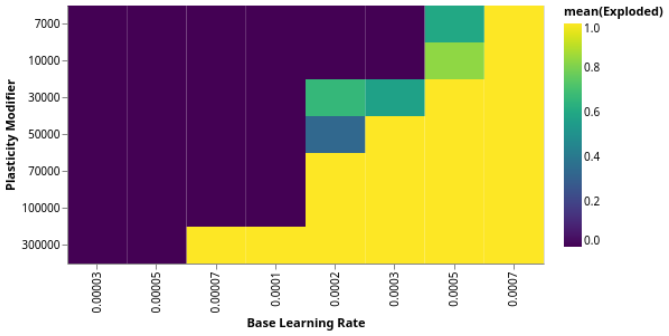

Fig. 9. Hyperparameter sweep results on the 3-character reversed sequence task showing the interaction between base learning rate and high-plasticity learning rate, 10 samples for each combination, on stability. A linear boundary separates runs that remained under a loss of 2 from those that underwent a runaway explosion of gradient values during a set number of training steps.

they also reduce memory duration below useful thresholds. Conversely, insufficient forgetting leads to inevitable instability. This creates a narrow operational window that varies significantly across tasks and datasets.

Synthetic datasets consistently exhibit more volatile behavior than natural text tasks, likely due to their non-smooth loss landscapes. The discrete, artificial nature of synthetic sequence tasks creates sharp loss cliffs that amplify the destabilizing effects of rapid weight changes, whereas natural language provides more gradual loss surfaces that better accommodate the ethereal weights mechanism.

3) Stabilization Strategies: Given the inherent volatility of ethereal weights, developing effective stabilization strategies becomes crucial for practical implementation. Our experiments reveal several approaches that can help balance stability with working memory effectiveness, though each comes with specific trade-offs.

Gradient Propagation Method Selection: Direct Feedback Alignment (DFA) emerges as the most effective approach for maintaining stability. Unlike standard backpropagation, which creates volatile feedback loops when plastic weights are used for both forward computation and gradient propagation, DFA employs fixed random feedback weights that remain constant throughout training. This architectural choice significantly reduces the positive feedback effects that lead to gradient explosion.

Weight Clipping: Constraining weight magnitudes through clipping proves highly effective when combined with standard backpropagation. By preventing weights from reaching extreme values, clipping helps contain the cascading effects that characterize gradient explosion. However, the same large weight magnitudes that threaten stability are often necessary for effective memory encoding. Overly aggressive clipping can therefore eliminate the model's working memory capabilities entirely, and turns out to be less useful when using DFA on more conservative choices of learning rate and ratio hyperparameters. A hyperparameter sweep on the clipping threshold was performed on the 3-character reversed sequence task, on selections of base learning rate and high-plasticity learning rate that were known to be on the edge of stability without clipping. The results were inconclusive, with 10 samples each on values between 0.01 and 100 showing no clear pattern of better stability or accuracy. This suggests that while weight clipping can help prevent explosion, its optimal setting is highly data-dependent.

Periodic Memory Reset: Wiping high-plasticity weights after sequence completion provides a hard reset mechanism that prevents the accumulation of destabilizing weight values across sequences. This approach proves particularly effective for tasks with clear sequence boundaries, essentially treating each sequence as an independent working memory episode. While this prevents cross-sequence contamination, it also eliminates any potential benefits from longer-term memory consolidation.

Ineffective Approaches: Several commonly used stabilization techniques prove counterproductive with ethereal weights. Gradient clipping, despite its effectiveness in traditional neural networks, fails because the memory mechanism fundamentally depends on occasional large gradient magnitudes to encode information effectively. Similarly, batch normalization and layer normalization actively harm performance by normalizing away the bimodal gradient distribution that enables memory encoding—the extreme gradient magnitudes these techniques

suppress are precisely what makes ethereal weights functional.

Dataset Choice: The choice of training data significantly impacts stability requirements. Natural language tasks, with their relatively smooth loss landscapes, suffer less readily from the volatile dynamics observed in this paper.

Extended Training Regimes: Some models exhibit U-shaped learning curves where initial gradient explosion eventually stabilizes after sufficient training. This phenomenon suggests that the optimization landscape contains stable regions accessible only after traversing unstable phases. While intriguing, this approach remains impractical for most applications due to the computational cost and unpredictability of the stabilization process.

Hyperparameter Co-optimization: The most promising long-term approach involves developing principled methods for jointly optimizing the base learning rate, plasticity modifiers, forgetting rates, and plasticity ratios. Current approaches rely heavily on task-specific hyperparameter sweeps, but future work should explore meta-learning approaches or adaptive mechanisms that can automatically balance stability and memory effectiveness across different tasks and datasets.

The fundamental challenge remains that ethereal weights operate in a narrow stability regime where effective memory encoding and system stability are in constant tension. Future implementations will likely require sophisticated control mechanisms that can dynamically adjust these parameters based on real-time monitoring of gradient ratios and other stability indicators.

4) Effective Learning Rate Paradox: An intriguing phenomenon emerges when calculating the effective learning rate across all parameters. Despite theoretical expectations, effective learning rates that would cause immediate explosion in traditional RNNs (often exceeding 20 in synthetic tasks) prove workable with ethereal weights. This paradox suggests that the forgetting mechanism fundamentally alters the learning dynamics, creating a different stability regime than conventional gradient descent.

However, this effective learning rate metric fails to correlate strongly with actual performance or stability, indicating that the temporal dynamics of forgetting introduce complexities not captured by simple averaging approaches. The ethereal component of the learning process appears to operate under different mathematical principles than standard parameter optimization.

D. Preservation of Long-Term Learning

A critical concern with the ethereal weights approach is whether the volatile behavior of high-plasticity weights interferes with the stable, long-term learning that should occur in the low-plasticity parameters. Our analysis demonstrates that these two learning processes can coexist effectively, with each operating at its designated timescale without mutual interference.

Fig. 10 reveals that gradient norms for high-plasticity and low-plasticity weights remain highly distinct throughout training, indicating that the two categories of parameters operate in separate dynamic regimes. The high-plasticity weights exhibit the expected volatility with occasional large spikes corresponding to memory encoding events, while the low-plasticity weights maintain steady, controlled gradient magnitudes characteristic of stable optimization.

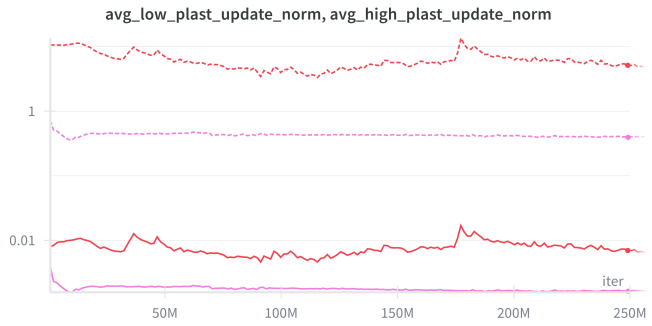

Fig. 10. Gradient norm evolution throughout training shows clear separation between high-plasticity and low-plasticity weight categories. The high-plasticity weights (red) exhibit characteristic volatility with occasional spikes corresponding to memory encoding events, while low-plasticity weights (blue) maintain steady, controlled gradient magnitudes. This separation demonstrates that the two learning processes operate independently, allowing stable long-term learning to proceed alongside volatile working memory encoding.

This separation is crucial for the dual-timescale learning hypothesis underlying ethereal weights. The steady decrease in loss observed across our experiments suggests that the model continues to learn long-term patterns at the low-plasticity learning rate while simultaneously managing working memory at the high-plasticity rate. This dual-process behavior indicates that ethereal weights successfully implement a form of multi-timescale learning where different parameter subsets serve distinct functional roles.

The preservation of long-term learning can be attributed to several factors. First, the random assignment of plasticity levels ensures that the network's fundamental computational capacity remains largely intact, with only a small fraction (typically 10-20%) of parameters dedicated to volatile memory functions. Second, the forgetting mechanism prevents high-plasticity weights from accumulating permanent biases that could interfere with the network's overall optimization trajectory.

To rigorously test this dual-learning hypothesis, future experiments could implement a controlled study where models are trained on a primary task (e.g., language modeling) while periodically exposed to working memory challenges (e.g., key-recall sequences). By measuring performance on both the primary task and working memory tasks throughout training, researchers could quantify how effectively the two learning processes coexist. Additionally, ablation studies could examine performance when high-plasticity weights are periodically frozen or reset, allowing direct measurement of the low-plasticity weights' learning progress in isolation.

Another promising experimental approach would involve training ethereal weights models on tasks with known long-term dependencies (such as copying tasks with very long delays) while simultaneously requiring short-term memory

operations. Success on both components would provide strong evidence that the architecture can maintain stable long-term learning while providing functional working memory capabilities.

The implications of this dual-timescale learning extend beyond the immediate applications of ethereal weights. This approach suggests a general framework for neural architectures that can simultaneously optimize for different temporal objectives, potentially offering new approaches to continual learning, meta-learning, and adaptive systems that must balance stability with flexibility.