Atlas: Learning to Optimally Memorize the Context at Test Time

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong,

Meisam Razaviyayn, and Vahab Mirrokni

Google Research

Abstract

Transformers have been established as the most popular backbones in sequence modeling, mainly due to their effectiveness in in-context retrieval tasks and the ability to learn at scale. Their quadratic memory and time complexity, however, bound their applicability in longer sequences and so has motivated researchers to explore effective alternative architectures such as modern recurrent neural networks (a.k.a long-term recurrent memory module). Despite their recent success in diverse downstream tasks, they struggle in tasks that requires long context understanding and extrapolation to longer sequences. We observe that these shortcomings come from three disjoint aspects in their design: (1) limited memory capacity that is bounded by the architecture of memory and feature mapping of the input; (2) online nature of update, i.e., optimizing the memory only with respect to the last input; and (3) less expressive management of their fixed-size memory. To enhance all these three aspects, we present Atlas, a long-term memory module with high capacity that learns to memorize the *context* by optimizing the memory based on the current and past tokens, overcoming the online nature of long-term memory models. Building on this insight, we present a new family of Transformer-like architectures, called DeepTransformers, that are strict generalizations of the original Transformer architecture. Our experimental results on language modeling, common-sense reasoning, recall-intensive, and long-context understanding tasks show that Atlas surpasses the performance of Transformers and recent linear recurrent models. Atlas further improves the long context performance of Titans, achieving +80% accuracy in 10M context length of BABILong benchmark.

1 Introduction

The attention module (Bahdanau et al. 2014) is a critical building block in modern deep learning architectures (Achiam et al. 2023; Behrouz, Zhong, et al. 2024; Kamath et al. 2025; Vaswani et al. 2017), excelling due to its scalability and performance in in-context retrieval tasks. In principle, attention functions as an associative memory, computing direct pairwise token dependencies to store key-value mappings and retrieve them via query-key similarities. Computing this pairwise dependencies, however, while accurate, causes quadratic space and time complexity, limiting their applicability in long context understanding, memorization, or modeling (Dalal et al. 2025; Li, Huang, et al. 2024; Liu, Lin, et al. 2024).

Recent research efforts aim to overcome the limitations of Transformers—i.e., pure attention-based architectures—in long-context modeling by designing more efficient yet effective recurrent neural networks (Behrouz, Zhong, et al. 2024; Peng, Zhang, et al. 2025; Schlag et al. 2021). These modern recurrent architectures can be unified as associative memory modules optimizing an internal objective termed 'attentional bias' (Behrouz, Razaviyayn, et al. 2025). Unlike Transformers' growing KV cache, these models use fixed-size memory, necessitating improved memory management. Consequently, there's growing interest in enhancing RNN memory management through more effective: (i) Learning rules, from additive learning (Katharopoulos et al. 2020) to DeltaNet's Delta rule (Schlag et al. 2021); (ii) Forget (Retention) Gates, from RetNet's input-independent gating (Sun, Dong, et al. 2023) to adaptive gating in Titans (Behrouz, Zhong, et al. 2024) and RWKV-7 (Peng, Zhang, et al. 2025); and (iii) Memory Architectures, from vector-valued memory (Peng, Alcaide, et al. 2023; Sun, Dong, et al. 2023) to neural deep memory modules (Behrouz, Zhong, et al. 2024; Sun, Li, et al. 2024).

Despite the success of these improved models in a diverse set of downstream benchmarks, they often struggle with long context understanding, in-context retrieval, and extrapolation to longer sequences (Arora, Eyuboglu, Zhang, et al. 2024;

[*] {alibehrouz, zemanli, pkacham, dengyuan, peilinz, razaviyayn, mirrokni}@google.com, and majiddl.2099@gmail.com

Table 1: A summary of the recent modern recurrent neural networks. We compare these architectures based on five characteristics: (1) Dynamic decay; (2) Deep neural memory; (3) non-linear memory capacity; (4) Locally optimal: managing memory by (approximating) the second-order information about tokens; (5) Flexible context: the ability to flexibly memorize the context. $\phi(\cdot)$ and $\phi^*(\cdot)$ represent polynomial and infinite-dimensional feature mappings (see Equation 22).

Model	Attentional Bias $\ell(\cdot;\cdot)$	Optimizer	Dynamic Decay	Deep Memory	Non-linear Capacity[†]	Locally Optimal	Flexible Context	Memory Write Operation
Attention	$\sum_{t=1}^{L} a_t \lVert \mathcal{M}\mathbf{k}_t - \mathbf{v}_t \rVert_2^2$	NP[‡]	✗	✗	✓	✓	✗	$\mathcal{M}_t = \mathcal{M}_{t-1} \cup \{(\mathbf{k}_t, \mathbf{v}_t)\}$
SWA	$\sum_{t=c}^{L} a_t \lVert \mathcal{M}\mathbf{k}_t - \mathbf{v}_t \rVert_2^2$	NP	✗	✗	✓	✓	✓	$\mathcal{M}_t = (\mathcal{M}_{t-1} \setminus \{(\mathbf{k}_c, \mathbf{v}_c)\}) \cup \{(\mathbf{k}_t, \mathbf{v}_t)\}$
Linear Attention	$\langle \mathcal{M}_t\mathbf{k}_t, \mathbf{v}_t \rangle$	GD	✗	✗	✗	✗	✗	$\mathcal{M}_t = \mathcal{M}_{t-1} + \mathbf{v}_t\mathbf{k}_t^\top$
RetNet	$\langle \mathcal{M}_t\mathbf{k}_t, \mathbf{v}_t \rangle$	GD	✗	✗	✗	✗	✗	$\mathcal{M}_t = \alpha\mathcal{M}_{t-1} + \mathbf{v}_t\mathbf{k}_t^\top$
GLA	$\langle \mathcal{M}_t\mathbf{k}_t, \mathbf{v}_t \rangle$	GD	✓	✗	✗	✗	✗	$\mathcal{M}_t = \mathrm{Diag}(\alpha_t)\mathcal{M}_{t-1} + \mathbf{v}_t\mathbf{k}_t^\top$
PolySketchFor.	$\langle \mathcal{M}_t\mathbf{k}_t^p, \mathbf{v}_t \rangle$	GD	✗	✗	✓	✗	✗	$\mathcal{M}_t = \mathcal{M}_{t-1} + \mathbf{v}_t(\mathbf{k}_t^\top)^p$
TTT	$\lVert \mathcal{M}_t(\mathbf{k}_t) - \mathbf{v}_t \rVert_2^2$	GD	✗	✓	✗	✗	✗	$\mathcal{M}_t = \mathcal{M}_{t-1} - \eta\nabla\ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t)$
DeltaNet	$\lVert \mathcal{M}_t\mathbf{k}_t - \mathbf{v}_t \rVert_2^2$	GD	✗	✗	✗	✗	✗	$\mathcal{M}_t = (\mathbf{I} - \beta_t\mathbf{k}_t\mathbf{k}_t^\top)\mathcal{M}_{t-1} + \beta_t\mathbf{v}_t\mathbf{k}_t^\top$
Longhorn	$\lVert \mathcal{M}_t\mathbf{k}_t - \mathbf{v}_t \rVert_2^2$	Implicit GD	✗	✗	✗	✗	✗	$\mathcal{M}_t = (\mathbf{I} - \delta_t\mathbf{k}_t\mathbf{k}^\top)\,\mathcal{M}_{t-1} + (\delta_t \odot \mathbf{v}_t)\,\mathbf{k}_t$ [§]
Gated DeltaNet	$\lVert \mathcal{M}_t\mathbf{k}_t - \mathbf{v}_t \rVert_2^2$	GD	✓	✗	✗	✗	✗	$\mathcal{M}_t = \alpha_t(\mathbf{I} - \beta_t\mathbf{k}_t\mathbf{k}_t^\top)\mathcal{M}_{t-1} + \beta_t\mathbf{v}_t\mathbf{k}_t^\top$
RWKV-7	$\lVert \mathcal{M}_t\mathbf{k}_t - \mathbf{v}_t \rVert_2^2$	GD	✓	✗	✗	✗	✗	$\mathcal{M}_t = (\mathrm{diag}(\alpha_t) - \beta_t\mathbf{k}_t\mathbf{k}_t^\top)\mathcal{M}_{t-1} + \beta_t\mathbf{v}_t\mathbf{k}_t^\top$
Titans	$\lVert \mathcal{M}_t(\mathbf{k}_t) - \mathbf{v}_t \rVert_2^2$	GD w/ M.[*]	✓	✓	✗	✗	✗	$\mathcal{M}_t = \alpha_t\mathcal{M}_{t-1} + \mathcal{S}_t$ $\mathcal{S}_t = \eta_t\mathcal{S}_{t-1} - \theta_t\nabla\ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t)$
Titans–	$\lVert \mathcal{M}_t(\mathbf{k}_t) - \mathbf{v}_t \rVert_2^2$	GD	✓	✓	✗	✗	✗	$\mathcal{M}_t = \alpha_t\mathcal{M}_{t-1} - \eta_t\nabla\ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t)$
Moneta	$\lVert \mathcal{M}_t(\mathbf{k}_t) - \mathbf{v}_t \rVert_p^p$	GD	✓	✓	✓	✗	✗	$\mathcal{M}_t = \alpha_t\mathcal{M}_{t-1} - \eta_t\nabla\ell(\mathcal{M}_{i-1}; \mathbf{k}_t, \mathbf{v}_t)$
Memora	$\lVert \mathcal{M}_t(\mathbf{k}_t) - \mathbf{v}_t \rVert_2^2$	GD	✓	✓	✗	✗	✗	$\mathcal{M}_t = \sigma\left(\alpha_t \log(\mathcal{M}_{t-1}) - \eta_t\nabla\ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t)\right)$
				Our Models				
DLA	$\langle \mathcal{M}_t(\phi(\mathbf{k}_t)), \mathbf{v}_t \rangle$	GD	✓	✓	✗	✗	✗	$\mathcal{M}_t = \alpha_t\mathcal{M}_{t-1} - \eta_t\nabla\ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t)$
DeepTransformer[⋇]	$\langle \mathcal{M}_t(\phi^*(\mathbf{k}_t)), \mathbf{v}_t \rangle$	GD	✓	✓	✓	✗	✗	$\mathcal{M}_t = \alpha_t\mathcal{M}_{t-1} - \eta_t\nabla\ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t)$
SWDT	$\sum_{i=c}^{L} \langle \mathcal{M}_t(\phi^*(\mathbf{k}_i)), \mathbf{v}_i \rangle$	GD	✓	✓	✓	✗	✓	$\mathcal{M}_t = \alpha_t\mathcal{M}_{t-1} - \eta_t\nabla\ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t)$
OmegaNet	$\sum_{i=c}^{L} \gamma_i \lVert \mathcal{M}_t(\phi(\mathbf{k}_i)) - \mathbf{v}_i \rVert_2^2$	GD	✓	✓	✓	✗	✓	$\mathcal{M}_t = \alpha_t\mathcal{M}_{t-1} - \eta_t\nabla\ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t)$
Dot[⋇]	$\sum_{i=c}^{L} \gamma_i \lVert \mathcal{M}_t(\phi^*(\mathbf{k}_i)) - \mathbf{v}_i \rVert_2^2$	GD	✓	✓	✓	✗	✓	$\mathcal{M}_t = \alpha_t\mathcal{M}_{t-1} - \eta_t\nabla\ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t)$
Atlas	$\sum_{i=c}^{L} \gamma_i \lVert \mathcal{M}_t(\phi(\mathbf{k}_i)) - \mathbf{v}_i \rVert_2^2$	Muon	✓	✓	✓	✓	✓	$\mathcal{M}_t = \alpha_t\mathcal{M}_{t-1} - \eta_t\ \texttt{NS-5}(\mathcal{S}_t)$ $\mathcal{S}_t = \theta_t\mathcal{S}_{t-1} - \nabla\ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t)$

[†] The matrix-valued memory version is considered. [‡] NP: Nonparametric [§] $\delta_t = \frac{\beta_t}{1+\beta_t\mathbf{k}_t^\top\mathbf{k}_t}$. [*] Gradient Descent with Momentum. [⋇] Without Normalization.

Behrouz, Zhong, et al. 2024; Wen et al. 2024; Yang, Kautz, et al. 2024). We observe these shortcomings arise from three design aspects: (1) The online nature of their memory update, where memory is optimized based on the current token while retaining past memory state, leading to memorization of individual tokens without considering broader context; (2) The limited capacity of memory, where architecture and key-value feature mappings restrict the number of perfectly mappable key-value pairs; and (3) The expressiveness of memory management (i.e., the internal objective's optimizer), as most recent models use gradient descent that relies on the first-order information about the dynamics of tokens, causing the memory to converge to spurious local minima and learn less effective key-value mappings.

Memory Perspective

Associative memory—the ability to map different entities or events—is an inseparable component of learning in humans (Terry 2017) and so has motivated several recent studies to understand the state-of-the-art deep learning architectures through its lens (Behrouz, Razaviyayn, et al. 2025; Behrouz, Zhong, et al. 2024; Ramsauer et al. 2021; Wang et al. 2025). In this perspective, memory is defined as a neural update caused by an input; the more surprising the input is, the more it affects the memory and so is memorable. Therefore, finding an effective "surprise metric" is a critical step towards designing such memory modules. As earlier discussed by Behrouz, Razaviyayn, et al. (2025) and Behrouz, Zhong, et al. (2024), almost all existing architectures use a surprise metric that updates the memory based on the current input. An event (as a sequence of tokens), however, might not consistently be surprising through a long-period of time although it is memorable. To overcome this issue, Behrouz, Zhong, et al. (2024) suggest breaking the surprise metric into two parts of "momentary" and "past" surprise, incorporating the *cumulative* surprise of past inputs when updating the memory with respect to the current input. This design, however, can miss the *context* by memorizing individual tokens. To this end, in this work, we present a long-term neural memory module that measures the surprise of a local (or global) context window, meaning that it learns how to memorize the (~~token~~) context at test time.

Through the paper, we use the terminology "Test Time Memorization" because the process involves storing and retrieving information strictly within the global context, without updating the model's core learned parameters (i.e., outer-loop) or initial states from pre-training. Typically, no persistent learning or skill acquisition carries over to new, independent global context once the memory is cleared. Thus, we prefer the use of "test time memorization" over using "test time training".

Contributions

In this paper, we aim to overcome the abovementioned limitations—i.e., (1) online nature, (2) limited memory capacity, and (3) less expressive memory management—by designing a long-term neural memory module with high capacity and the ability to memorize the context, instead of tokens. We further build upon these insights and present a family of strictly more powerful Transformers. More specifically:

Better Understanding of Memory Capacity and its Bottleneck. To improve the limited memory capacity, we suggest using higher-order feature mappings (e.g., polynomial feature kernels) on input tokens. We provide theoretical justifications on why deeper memory modules and/or higher-order feature mapping can enhance memory capacity—i.e., the maximum number of linearly independent key-value associations the memory can perfectly map.

New Expressive Learning Rule. To overcome the online nature of recent recurrent models, this work presents a sliding window update rule, called Omega rule, that optimizes and updates memory based on all past tokens in a given context window, not just the last. This allows the model to better manage its fixed-size memory and memorize a local context instead of individual tokens.

Strict Generalization of Transformers. Next, we show how our Omega rule formulation connects to global and local softmax attentions (i.e., Sliding Window Attention - SWA) and present a new family of Transformer-like architectures, called DeepTransformers and its sliding window variants SWDT, that strictly generalize Transformers (Vaswani et al. 2017). We further present a novel baseline of Deep Linear Attention (DLA) to demonstrate the role of deep memory.

New Memory Modules with Better Memory Management. Building upon the above improvements, we present OmegaNet, a new architecture using polynomial features on its keys and queries, while updating its memory based on Omega and gradient descent. To further enhance memory management, we introduce Atlas, which leverages the popular Muon optimizer (Jordan et al. 2024) for updating the internal memory. We show that both OmegaNet and Atlas can take advantage of parallelizable training algorithms, resulting in fast training without substantial overhead compared to the online version (i.e., context window = 1). To the best of our knowledge, Atlas is the first parallelizable recurrent architecture that optimizes the memory using the (approximation) of second-order information (i.e., has locally optimal memory module).

Improvement on Diverse Downstream Tasks. Extensive experiments validate our model designs and proposed techniques, including ablations of modern architectures. We evaluated DeepTransformers, OmegaNet, and Atlas on diverse benchmarks—language modeling, common-sense reasoning, recall-intensive, and needle-in-haystack tasks—where they outperformed modern linear RNNs, local attention (SWA), and Transformers. Furthermore, we studied the effects of memory architecture, feature mapping, memory management algorithm (internal optimizer), and Omega rule on memory module capacity and performance in long-context understanding tasks.

Proofs, additional experimental results, discussions on related work, and the details of experiments are in Appendix.

2 Preliminaries

In this section, we first discuss the notation that we use through the paper and then review the background concepts and related work. Additional discussion on related studies are in Appendix A.

Notations. We let $x \in \mathbb{R}^{N \times d_{\text{in}}}$ be the input, $\mathcal{M}_t$ be the state of memory $\mathcal{M}$ at time t, $\mathbf{K}$ be the keys, $\mathbf{V}$ be the values, and $\mathbf{Q}$ be the query matrices. We use bold lowercase letters with subscript t to refer to vectors correspond to time t (i.e., $\mathbf{k}_t, \mathbf{v}_t$, and $\mathbf{q}_t$). Following Behrouz, Razaviyayn, et al. (2025), we use $\ell(\mathcal{M}_t; \mathbf{k}_t, \mathbf{v}_t)$ to refer to the attentional bias (i.e., the internal memory objective). Through the paper, we use simple MLPs with $\mathcal{L}_{\mathcal{M}} \geq 1$ layers and residual connection as the architecture of the memory module $\mathcal{M}(\cdot)$. Notably, despite this choice, all of our model formulations are simply adaptable to other memory architecture choices; e.g., linear matrix-valued memory ($\mathcal{L}_{\mathcal{M}} = 1$). When it is needed, we parameterized the memory module with $\boldsymbol{\theta}_{\mathcal{M}} := \{W_1, \dots, W_{\mathcal{L}_{\mathcal{M}}}, \dots\}$, which at least includes the parameters of linear layers in the MLP.

2.1 Backgrounds

Attention. Attention is a critical component of Transformers that acts as their associative memory (Behrouz, Razaviyayn, et al. 2025; Bietti et al. 2023; Sun, Li, et al. 2024). Given input $x \in \mathbb{R}^{N \times d_{\text{in}}}$, causal attention computes output $\mathbf{y} \in \mathbb{R}^{N \times d_{\text{in}}}$ over input dependent key, value, and query matrices $\mathbf{Q} = x\mathbf{W}_{\mathbf{Q}}, \mathbf{K} = x\mathbf{W}_{\mathbf{K}},$ and $\mathbf{V} = x\mathbf{W}_{\mathbf{V}}$ as:

$$\mathbf{y}_i = \sum_{j=1}^{i} \frac{\exp\left(\mathbf{q}_i^\top \mathbf{k}_j / \sqrt{d_{\text{in}}}\right) \mathbf{v}_j}{\sum_{\ell=1}^{i} \exp\left(\mathbf{q}_i^\top \mathbf{k}_\ell / \sqrt{d_{\text{in}}}\right)} = \frac{1}{Z_i} \sum_{j=1}^{i} \exp\left(\mathbf{q}_i^\top \mathbf{k}_j / \sqrt{d_{\text{in}}}\right) \mathbf{v}_j, \tag{1}$$

where $\mathbf{W}_{\mathbf{Q}}, \mathbf{W}_{\mathbf{K}},$ and $\mathbf{W}_{\mathbf{V}} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$ are learnable parameters, and $Z_i = \sum_{\ell=1}^{i} \exp\left(\mathbf{q}_i^\top \mathbf{k}_\ell / \sqrt{d_{\text{in}}}\right)$ is the normalization term. Despite Transformers' simple parallelizable training and effectiveness in recall-intensive tasks (Arora, Eyuboglu, Zhang, et al. 2024), their generation process and long-context scaling are significant drawbacks, as attention requires at least $N \times d$ operations per token to calculate the output (see Equation 1). Therefore, in recent years, there have been an extensive research effort to design alternative architectures. We divide and review these studies into two groups: (1) Linear shallow memory recurrent models, (2) Deep memory modules:

(Linear) Recurrent Models. Linear RNNs have recently gained attention as efficient Transformer alternatives due to their parallelizable, linear-time training and comparable performance (Peng, Alcaide, et al. 2023; Sun, Dong, et al. 2023). Early modern RNN variants, often based on Hebbian (Hebb 2005) or Delta (Widrow et al. 1988) learning rules, compress data into vector-valued or matrix-valued memory (Kacham et al. 2024b; Katharopoulos et al. 2020; Lim et al. 2024; Liu, Wang, et al. 2024; Schlag et al. 2021; Sun, Dong, et al. 2023). Let $\mathcal{M}_t \in \mathbb{R}^{d \times n}$ be the memory (where $n = 1$ yields vector-valued memory), and $\mathbf{k}, \mathbf{v} \in \mathbb{R}^d$ be the keys and values (projections of input $x_t \in \mathbb{R}^d$)). A simple general formulation for such linear RNNs is:

$$\mathcal{M}_t = A_t * \mathcal{M}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top, \tag{2}$$

where $*$ is an arbitrary associative operator and A_t is a data-(in)dependent diagonal or low-rank plus identity matrix (Yang, Wang, Zhang, et al. 2024). Despite the efficient *linear* recurrent nature of these models, their memory can overflow, particularly with increasing context length. Although forget gates have recently significantly improved memory management in these architectures (Peng, Zhang, et al. 2025; Sun, Dong, et al. 2023), their memory's expressivity remains bounded by its linear structure.

Deep Memory Module. To overcome the limited expressivity of memory and to enhance the *effective* context length recurrent models, recent studies focus on a new line of architectures with deep memory modules (Behrouz, Razaviyayn, et al. 2025; Behrouz, Zhong, et al. 2024; Irie et al. 2021; Sun, Li, et al. 2024). These architectures are built on the meta-learning perspective, where the memory is a deep MLP architecture updated by gradient descent (with momentum). Recently, Behrouz, Razaviyayn, et al. (2025) present a framework to accurately unifies popular sequence models as the instances of test time memorization. That is, sequence models are associative memory modules that aim to learn the underlying mapping between given keys and values by optimizing an internal memory objective, called attentional bias. This optimization is based on an iterative optimization algorithms such as gradient descent. More formally, associative memory is defined as:

Definition 1 (Behrouz, Razaviyayn, et al. (2025))**.** *Given a set of keys $\mathcal{K} \subseteq \mathbb{R}^{d_k}$ and values $\mathcal{V} \subseteq \mathbb{R}^{d_v}$, associative memory is an mapping $\mathcal{M} : \mathcal{K} \to \mathcal{V}$. Learning the associative memory is based on an objective $\mathcal{L}$, called* Attentional Bias*, that determines the type of memory and its priorities:*

$$\mathcal{M}^* = \arg\min_{\mathcal{M}} \quad \mathcal{L}(\mathcal{M}(\mathcal{K}); \mathcal{V}). \tag{3}$$

Optimizing this objective using an iterative algorithm (e.g., gradient descent) results in the memory update rule. Thus, the sequence model is a meta in-context learner with two optimization levels:

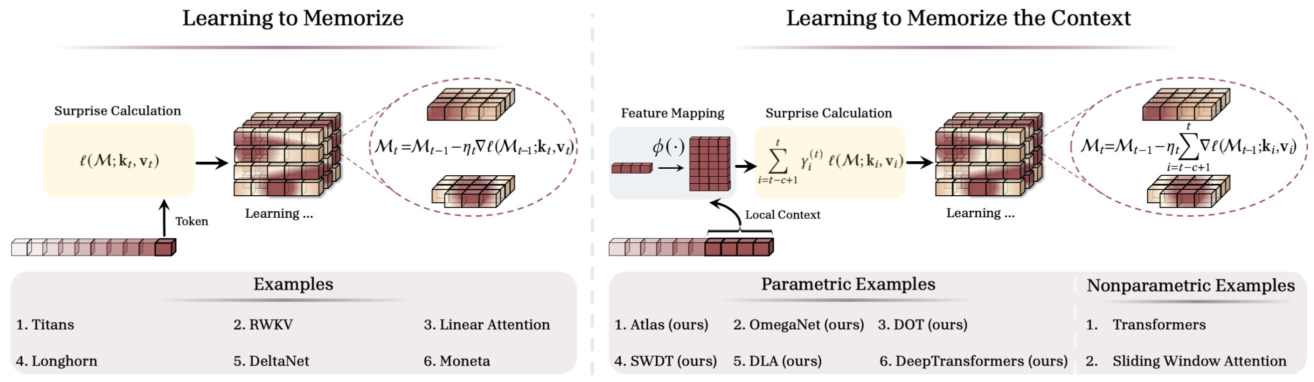

Learning to Memorize
Surprise Calculation
$\ell(\mathcal{M}; \mathbf{k}_t, \mathbf{v}_t)$
Token
Learning ...
$\mathcal{M}_t = \mathcal{M}_{t-1} - \eta_t \nabla \ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t)$
Examples
1. Titans
2. RWKV
3. Linear Attention
4. Longhorn
5. DeltaNet
6. Moneta
Learning to Memorize the Context
Feature Mapping
$\phi(\cdot)$
Surprise Calculation
$\sum_{i=t-c+1}^{t} \gamma_i^{(t)} \ell(\mathcal{M}; \mathbf{k}_i, \mathbf{v}_i)$
Local Context
Learning ...
$\mathcal{M}_t = \mathcal{M}_{t-1} - \eta_t \sum_{i=t-c+1}^{t} \nabla \ell(\mathcal{M}_{t-1}; \mathbf{k}_i, \mathbf{v}_i)$
Parametric Examples
1. Atlas (ours)
2. OmegaNet (ours)
3. DOT (ours)
4. SWDT (ours)
5. DLA (ours)
6. DeepTransformers (ours)
Nonparametric Examples
1. Transformers
2. Sliding Window Attention

Figure 1: Comparison of learning to memorize (**Left**) individual tokens, and (**Right**) the context.

1. Inner Loop: Where parameters of the memory module are optimized (i.e., $\boldsymbol{\theta}_{\mathcal{M}} = \{W_1, W_2, \dots, W_{\mathcal{L}_{\mathcal{M}}, \dots}\}$). In the inner optimization loop, all other parameters from the model are considered hyperparameters and are fixed and *not* optimized.
2. Outer Loop: Where all other parameters of the model are optimized, such as linear projections, MLP layers, convolutions, etc.

Our terminology builds on this framework. Therefore, instead of full recurrent formulations, we describe models by their: (1) memory architecture, (2) internal objective (i.e., attentional bias), and (3) memory learning algorithm (optimizer). In most cases, models use matrix-valued memory with online gradient descent; for brevity in such instances, we refer to an architecture solely by its internal memory objective. For additional discussions and examples, see Appendix B.

3 Learning to Memorize the Context at Test Time

Long-term associative memory, crucial for human learning (Terry 2017), has inspired many artificial neural architectures (Behrouz, Razaviyayn, et al. 2025; Behrouz, Zhong, et al. 2024; He et al. 2024; Hopfield 1982; Krotov and Hopfield 2016; Ramsauer et al. 2021; Schmidhuber and Hochreiter 1997). While many such models use matrix- or vector-valued memory to compress past data (Schlag et al. 2021; Von Oswald et al. 2023; Yang, Kautz, et al. 2024), recent studies advocate for deep non-linear neural memory that encodes past abstractions into its parameters (Behrouz, Razaviyayn, et al. 2025; Behrouz, Zhong, et al. 2024; Dalal et al. 2025; Sun, Li, et al. 2024). For long-context reasoning/understanding, however, these long-term neural memory modules still require: (1) High capacity—the maximum (key, value) pairs storable in parameters (see §3.1); (2) A powerful internal memory objective (i.e., *attentional bias*) to learn complex mapping between keys and values (see §3.2); (3) Powerful memory management for better fixed-size memory management (see §3.2); and (4) An efficient parallel training process for large-scale training on modern accelerators (see §3.3).

This section further discusses these challenges and presents Omega rule: an expressive memory update rule with direct access to tokens in a local context window, which memorizes context rather than individual tokens.

3.1 Associative Memory with Super Linear Capacity

As previously discussed, an effective long-term memory module should store past data abstractions in its parameters. However, with a fixed number of memory parameters, a key unanswered question remains: "*what is the maximum number of uncorrelated (key, value) pairs that a model can store?*" To answer this, we start with the simplest case: matrix memory, an ℓ_2 regression loss as the attentional bias (i.e., $\ell(\mathcal{M}_t; \mathbf{k}_t, \mathbf{v}_t) = \|\mathcal{M}_t(\mathbf{k}_t) - \mathbf{v}_t\|_2^2$), optimized by gradient descent:

Proposition 1 (Capacity of ℓ_2 Attentional Bias). *Let $\mathcal{M}$ be a matrix-valued memory with $d_v \times d_k$ parameters that optimizes the internal objective of $\ell(\mathcal{M}_t; \mathbf{k}_t, \mathbf{v}_t) = \|\mathcal{M}_t \mathbf{k}_t - \mathbf{v}_t\|_2^2$ with gradient descent. $\mathcal{M}$ can store the mapping of at most $O(d_k)$ pairs of $(\mathbf{k}_i, \mathbf{v}_i)$ with linearly independent keys.*

The above proposition indicates that matrix-valued memory with delta update rule has sub-linear capacity with respect to its number of parameters. This means that the number of independent patterns that can be stored in a fixed-size memory with size M is strictly less than $c \times M$, for some $c \in \mathbb{R}^+$. Recent recurrent models suggest using deep memory modules to store the abstraction of the past into the parameters of a deep neural network (Behrouz, Razaviyayn, et al. 2025; Behrouz, Zhong, et al. 2024; Irie et al. 2021; Sun, Li, et al. 2024). While these deep memory architectures can intuitively enhance the expressive power in modeling complex underlying mapping patterns between keys and values, it is still unclear that if they enhance the memory capacity.

Theorem 1 (Effect of Deep Memory). *Let $\mathcal{M}(\cdot)$ be an MLP with $\mathcal{L}_{\mathcal{M}} \geq 2$ layers, d_k input dimension, and d_h hidden dimension. Then, $\mathcal{M}(\cdot)$ can store the mapping of at least $O(d_k d_v)$ and at most $O\left(d_k d_v \sum_{i=1}^{\mathcal{L}_{\mathcal{M}}} \min\{d_h^{(j)}\}_{j \geq i} d_h^{(j+1)}\right)$ pairs of $(\mathbf{k}_i, \mathbf{v}_i)$ with linearly independent keys.*

This theorem indicates that deep memory not only improves representational power but also further boosts network capacity, with advantages growing with depth. However, the upper bound remains subquadratic in key and value dimensions, raising the question if a long-term memory module can achieve super-linear capacity.

As stated earlier, the dimension of $\mathbf{k}_t$s is crucial for increasing memory capacity. Simply increasing all key and value dimensions, however, significantly increase the number of parameters ($O(d_{\text{in}})$ per each extra dimension) and memory usage, particularly with long contexts. To address this, building on methods from Kacham et al. (2024a) and Krotov and Hopfield (2016), we suggest using separable kernels $\sigma(x, y) = \phi(x)^\top \phi(y)$ for keys and queries. As an example of such kernels, we focus on polynomial kernels of degree at most p to increase input dimensionality and thus network capacity. Given $p \in \mathbb{N}$, let $\phi_p(x) = [x^\beta]_{|\beta| \leq p}$ be a polynomial mapping of x with degree at most p. We redefine the associative memory module in Definition 1 by replacing the inner objective of $\mathcal{L}(\mathcal{M}(\mathcal{K}); \mathcal{V})$ with $\mathcal{L}(\mathcal{M}(\phi(\mathcal{K})); \mathcal{V})$. This polynomial mapping enhances representational power by increasing the effective dimensionality of keys without additional parameter overhead for the input projections. Next, we discuss their effect on memory capacity, even with a single matrix-valued memory:

Proposition 2 (Memory Capacity with Polynomial Mapping). *Let $\phi_p(\cdot)$ be a polynomial mapping with degree at most p, and $\mathcal{M}$ be a matrix-valued memory that optimizes the internal objective of $\ell(\mathcal{M}_t; \phi_p(\mathbf{k}_t), \mathbf{v}_t) = \|\mathcal{M}_t \phi_p(\mathbf{k}_t) - \mathbf{v}_t\|_2^2$ with gradient descent. $\mathcal{M}$ can store the mapping of at most $O\left(d_k{}^p\right)$ pairs of $(\mathbf{k}_i, \mathbf{v}_i)$ with linearly independent keys, where d_k is the dimension of keys $\mathbf{k}_i$.*

Beyond the above intuition, polynomial kernels are further motivated by two perspectives: (1) Approximating Softmax using Taylor series; and (2) Input feature gating. For the sake of clarity, we continue with linear memory and two popular attentional biases i.e., $\ell^{(1)}(\mathcal{M}_t; \mathbf{k}_t, \mathbf{v}_t) = \langle \mathcal{M}_t \mathbf{k}_t, \mathbf{v}_t \rangle$ and $\ell^{(2)}(\mathcal{M}_t; \mathbf{k}_t, \mathbf{v}_t) = \|\mathcal{M}_t \phi(\mathbf{k}_t) - \mathbf{v}_t\|_2^2$. The same process can be applied on other attentional objectives and deep memory modules. Optimizing these objectives using gradient descent in the inner loop results in the following recurrent formulas:

$$\ell^{(1)}(\mathcal{M}_t; \mathbf{k}_t, \mathbf{v}_t) \quad : \quad \mathcal{M}_t = \mathcal{M}_{t-1} + \eta_t \mathbf{v}_t \phi(\mathbf{k}_t)^\top, \qquad \text{(Hebbian Rule)}$$

$$\ell^{(2)}(\mathcal{M}_t; \mathbf{k}_t, \mathbf{v}_t) \quad : \quad \mathcal{M}_t = \left(\mathbf{I} - \eta_t \phi(\mathbf{k}_t) \phi(\mathbf{k}_t)^\top\right) \mathcal{M}_{t-1} + \eta_t \mathbf{v}_t \phi(\mathbf{k}_t)^\top. \qquad \text{(Delta Rule)}$$

Kernel Attention Perspective for the Special Case of Hebbian Rule. The formulation for (Hebbian Rule) is equivalent to kernel linear attentions (Arora, Eyuboglu, Zhang, et al. 2024; Hua et al. 2022; Kacham et al. 2024b; Kasai et al. 2021; Katharopoulos et al. 2020; Wang et al. 2025). In this viewpoint, the role of $\phi(.)$ is to approximate `Softmax` or more accurately the exponential kernel. Since exponential kernel with normalization (i.e., `Softmax`) is not separable, it results in Transformers' quadratic time and memory complexity. However, Transformers' exponential feature map kernel ($\exp(\mathbf{q}_i^\top \mathbf{k}_j)$) can be approximated using its Taylor series as:

$$\exp\left(\mathbf{q}_i^\top \mathbf{k}_j\right) \approx 1 + \mathbf{q}_i^\top \mathbf{k}_j + \frac{(\mathbf{q}_i^\top \mathbf{k}_j)^2}{2!} + \frac{(\mathbf{q}_i^\top \mathbf{k}_j)^3}{3!} + \dots \tag{4}$$

Our polynomial feature map extends this approximation to a more general case of:

$$\exp\left(\mathbf{q}_i^\top \mathbf{k}_j\right) \approx \phi_p(\mathbf{q}) \phi(\mathbf{k})^\top = a_0 + a_1 \mathbf{q}_i \mathbf{k}_j^\top + a_2 (\mathbf{q}_i^\top \mathbf{k}_j)^2 + a_3 (\mathbf{q}_i^\top \mathbf{k}_j)^3 + \cdots + a_p (\mathbf{q}_i^\top \mathbf{k}_j)^p, \tag{5}$$

with learnable parameters $a_i \in \mathbb{R}$ initialized at $a_i = \frac{1}{i!}$, the polynomial kernel can be viewed as an expressive approximator of `Softmax` attention. This provides theoretical motivation for using polynomial kernels, especially when memory capacity is limited; i.e., with (i) linear memory and (ii) Hebbian learning rule. This intuition, however, further generalizes to more expressive cases using deep memory modules and more complex attentional biases (i.e., Eq. Delta Rule). That is, $\exp(\cdot)$ feature mapping has infinite dimension and provides a more powerful similarity measure of keys and queries (i.e., $\mathbf{q}_i^\top \mathbf{k}_j$);

however, its computation with normalization can cause additional memory and time complexity to the model. Using polynomial kernels in architectures with deep memory and complex attentional bias can further enhance performance by approximating more powerful representations for keys-queries similarities (i.e., $\mathbf{q}_i^\top \mathbf{k}_j$). See Section 4 for additional discussions on exponential kernels and Transformers.

Input Gating Interpretation. Another perspective that motivates the use of polynomial features is their more expressive representational power in modeling complex functions compared to the simple case of $\phi(x) = x$. That is, the coefficients of a_is can be seen as input feature gating, in which $a_i \longrightarrow 0$ means excluding the feature map of $[x^j]_{|j|=i}$, and $a_i \longrightarrow 1$ means retaining the corresponding feature. This is similar to the gating mechanisms of RNNs but on the input rather than the memory. This gating mechanism clearly provides a more representational power as the model can learn to set $a_i \longrightarrow 0$ for all $i \neq 1$ and $a_1 \longrightarrow 1$, resulting in the simple case of $\phi(x) = x$.

3.2 Long-term Memory with Context Memorization

As discussed earlier, one of the critical drawback of most existing recurrent models is their online nature, in which they optimize the inner objective (attentional bias) with respect to only the current input while retaining the previous state of the memory (Behrouz, Razaviyayn, et al. 2025; Liu, Wang, et al. 2024), i.e.,

$$\min_{\mathcal{M}} \ell(\mathcal{M}; \mathbf{k}_t, \mathbf{v}_t) + \texttt{Ret}_t(\mathcal{M}, \mathcal{M}_{t-1}), \tag{6}$$

where $\texttt{Ret}(\cdot, \cdot)$ is the retention gate. This online nature while making the optimization of the memory simpler and faster, can cause sub-optimal memorization of the context as memory is greedily memorize individual tokens. In a more general case, however, one can optimize the memory at each time stamp with respect to the entire context (input sequence), i.e.,

$$\min_{\mathcal{M}} \sum_{i=1}^{t} \ell(\mathcal{M}; \mathbf{k}_i; \mathbf{v}_i). \tag{7}$$

This strict global formulation generally presents two critical limitations: (1) Efficiency: One of the important advantages of recurrent architectures is their efficiency at longer context in both training and inference. Optimizing the memory with respect to all the past tokens (entire context), however, (i) causes additional optimization constraints at each memory update step, resulting in inefficiency at extremely large sequences, and (ii) requires caching the past keys and values at the test time, increasing the memory consumption; (2) Context Pruning: In large context tasks optimizing with all past tokens can cause sub-optimal performance mainly due to the context change (or irrelevant context) in the middle of the input sequence. This observation has resulted to design architectures with retention (forget) gate, enabling models to erase memory when past context is no longer needed (Behrouz, Razaviyayn, et al. 2025; Behrouz, Zhong, et al. 2024; Peng, Zhang, et al. 2025; Sun, Dong, et al. 2023; Yang, Wang, Shen, et al. 2024).

To address these limitations, we present a sliding window recurrent model that optimizes its attentional bias w.r.t. a window of past tokens. For a memory module $\mathcal{M}(\cdot)$ and window length $c \geq 1$, we optimize the memory internal objective as:

$$\min_{\mathcal{M}} \sum_{i=t-c+1}^{t} \gamma_i^{(t)} \, \ell(\mathcal{M}; \mathbf{k}_i, \mathbf{v}_i), \tag{8}$$

where $\ell(\mathcal{M}; \mathbf{k}_i, \mathbf{v}_i)$ measures the predicted mapping for $(\mathbf{k}_i, \mathbf{v}_i)$ pair and $\gamma_i^{(t)}$ is the decay term for the effect of i-th token in the optimization process. Building upon this formulation, we present Omega rule, which is strictly more powerful than the popular Delta learning rule (Schlag et al. 2021; Widrow et al. 1988):

Omega Rule: Let $\mathbf{k}_i \in \mathbb{R}^{d_k}$ and $\mathbf{v}_i \in \mathbb{R}^{d_v}$ be the input keys and values, and $\mathcal{M}(\cdot)$ be a neural architecture that serves as the memory module. Given a local context length of $c \in \mathbb{N}_{\geq 1}$, the updating the memory module $\mathcal{M}$ using Omega learning rule is defined as optimizing the following loss function with gradient descent:

$$\min_{\mathcal{M}} \sum_{i=t-c+1}^{t} \gamma_i^{(t)} \left\| \mathcal{M}\left(\mathbf{k}_i\right) - \mathbf{v}_i \right\|_2^2 \tag{9}$$

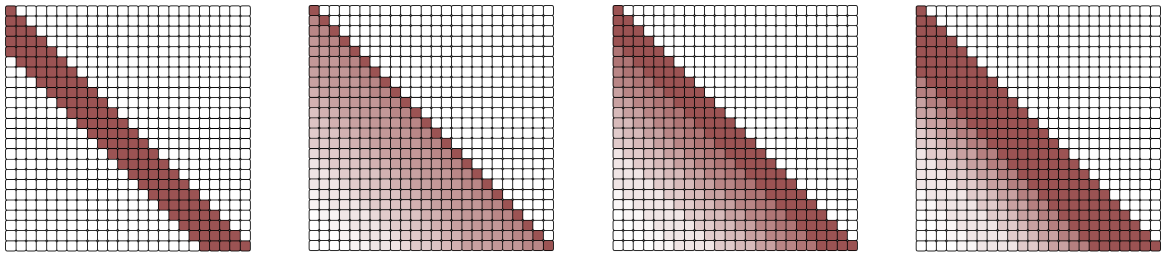

Sliding Window Attention
ATLAS (Context Window = 1)
ATLAS (Context Window = 4)
ATLAS (Context Window = 7)

Figure 2: The illustration of tokens dependencies in SWA and Atlas or OmegaNet with different context length.

Following Behrouz, Razaviyayn, et al. 2025, this update rule can be extended to q-Omega rule (or other variants) by replacing $\ell_2(\cdot)$ with $\ell_q(\cdot)$. In the extreme cases of (1) $c = 1$: the update rule becomes online (Delta rule); and (2) $c = \infty$ or context length: the update becomes global optimization w.r.t. all past tokens. In this formulation, parameters $\gamma_i^{(t)} \in [0, 1]$ act as hard (direct) gates for the past tokens. That is, $\gamma_i^{(t)} \longrightarrow 0$ means that the model directly prunes the optimization of i-th token in the local context, while $\gamma_i^{(t)} \longrightarrow 1$ means fully incorporating the optimization of memory for i-th token in the local context. In our design, we use input-dependent parameters for $\gamma_i^{(t)}$, providing in-context pruning ability. Note that, the design of sliding window recurrence allows such flexibility as for each token we need a constant number of gates; i.e., $\{\gamma_i^{(t)}\}_{i=1}^{c}$. Using input-dependent gates for the global optimization (Equation 7), however, can result in significant parameter increase and memory usage, diminishing the advantages of recurrent models.

OmegaNet. We now present OmegaNet, a novel sequence model that updates its memory using Omega rule. To enhance the memory capacity of OmegaNet, we use polynomial kernels on $\mathbf{k}$s and $\mathbf{q}$s. Accordingly, optimizing the objective in Equation 9, results in an update rule of OmegaNet as:

$$\mathcal{M}_t = \alpha_t \mathcal{M}_{t-1} - \underbrace{\nabla \sum_{i=t-c+1}^{t} \gamma_i^{(t)} \left\| \mathcal{M}\left(\phi(\mathbf{k}_i)\right) - \mathbf{v}_i \right\|_2^2}_{\text{Surprise of the context}}, \tag{10}$$

or in the spacial case of linear memory:

$$\mathcal{M}_t = \left(\texttt{diag}(\alpha_t) - \sum_{i=t-c+1}^{t} \gamma_i^{(t)} \phi(\mathbf{k}_i) \phi(\mathbf{k}_i)^\top \right) \mathcal{M}_{t-1} - \sum_{i=t-c+1}^{t} \gamma_i^{(t)} \mathbf{v}_i \phi(\mathbf{k}_i)^\top. \tag{11}$$

From the memory perspective, Omega rule (OmegaNet) does not measure the surprise of a token, but the surprise of a local context based on the context-aware combination of individual tokens within the context.

Beyond Gradient Descent. The concept of Omega rule and "test time memorization of context" can simply be extended to optimizing the objective in Equation 9 with any arbitrary optimizer, even beyond simple gradient descent. We use two extreme cases for c as the illustrations. In the first case, we let $c = 1$, $\gamma_i^{(t)} = 1$, and use gradient descent with momentum as the optimizers, resulting in the following update rule:

$$\mathcal{M}_t = \alpha_t \mathcal{M}_{t-1} + \mathcal{S}_t \tag{12}$$
$$\mathcal{S}_t = \theta_t \mathcal{S}_{t-1} - \eta_t \nabla \ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t). \tag{13}$$

This update rule is equivalent to the long-term neural memory in Titans (Behrouz, Zhong, et al. 2024). In the second case, using a linear memory $\mathcal{M}$, letting $\gamma_i^{(t)} = 1$, and c be equal to the context length, the memory update process is equivalent to optimizing the (regularized) least-squares problem:

$$\mathcal{M}_t = \min_{\mathcal{M}} \sum_{i=1}^{t} \left\| \mathcal{M}\mathbf{k}_i - \mathbf{v}_i \right\|_2^2 . \tag{14}$$

Von Oswald et al. (2023) suggest directly optimizing the above objective and use Sherman-Morrison formula (Sherman et al. 1950) to recursively calculate the inverse term in the optimal solution. Despite the optimality of memory, such direct solutions comes with the cost of non-parallelizable training and also are limited to only the linear matrix-valued memory setup. Furthermore, as discussed earlier, the global nature without any direct hard gating terms (i.e., $\gamma_i^{(t)}$s) can force the model to *not* prune the context, damaging the performance in longer sequences.

3.3 Parallelizing Omega Rule

While Omega rule provides a more general and expressive formulation for the design of memory modules than Hebbian or Delta learning rules, its applicability to large-scale models relies on its efficiency in training. To this end, we discuss a fast parallelizable training algorithms that does not add any significant computational overhead with the online counterpart version (i.e., $c = 1$). A naive implementation requires materializing c gradients $\nabla \ell \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$, which can result in a significantly higher memory footprint and I/O cost when d_{in} is large. Also, to fully utilize hardware accelerators such as TPUs and GPUs, it is important to tensorize computations and maximize the use of `matmul` operations. Motivated by recent work (Behrouz, Zhong, et al. 2024; Sun, Li, et al. 2024), we propose a simple sliding window masking strategy that supports efficient parallel training while avoiding substantial memory overhead. Specifically, we partition the input sequence with length L into chunks of size $b \geq 1$, each of which is represented by $\mathbf{S}_i = \{\mathbf{x}_{(i-1)b+1}, \dots, \mathbf{x}_{ib}\}$. Then for each chunk, we calculate the gradients with respect to the last state of the previous chunk. For the sake of clarity, we first assume $\gamma_i^{(t)} = \eta_t$ for all positions in the sequence. When the chunk size is $b = 1$, the update rule is:

$$\mathcal{M}_t = \alpha_t \mathcal{M}_{t-1} - \eta_t \sum_{i=t-c+1}^{t} \nabla \ell(\mathcal{M}_{t-1}; \mathbf{k}_i, \mathbf{v}_i), \tag{15}$$

where $\mathcal{M}_t$ is the model state at step t, α_t and η_t are the weight decay and learning rate parameters respectively, and $(\mathbf{k}_i, \mathbf{v}_i)$ denote the input pair at position i. In practice, we strike a balance between the fully recurrent form and the fully parallel form by dividing the sequence into smaller chunks. Within each chunk (intra-chunk), we apply parallel computation, while across chunks (inter-chunk), we adopt a recurrent computation scheme. We now define $t' = t - \text{mod}(t, b)$. That is, for time steps t such that $t' \leq t < t' + b$, the update rule within each chunk becomes:

$$\mathcal{M}_t = \alpha_t \dots \alpha_{t'} \mathcal{M}_{t'} - \sum_{n=t'}^{t} \frac{\alpha_t \dots \alpha_{t'}}{\alpha_n \dots \alpha_{t'}} \eta_n \underbrace{\sum_{i=n-c+1}^{n} \nabla \ell(\mathcal{M}_{t'}; \mathbf{k}_i, \mathbf{v}_i)}_{G_t} \tag{16}$$

In our implementation, for G_t, we follow the same gradient computation approach as described in Titans (Behrouz, Zhong, et al. 2024) but additionally apply a sliding window mask M_s during the broadcasting operation (e.g., using `einsum`). When $c = 1$, the sliding window mask M_s reduces to the identity matrix. For $c > 1$, M_s is an identity matrix except that the $c - 1$ positions immediately preceding each diagonal entry are also set to 1. This allows gradient contributions from a window of size c, enabling efficient computation without materializing all gradients inside the chunk.

4 DeepTransformers: Transformers with Deep Memory

Recent studies have extensively discussed Transformer architectures through the lens of associative memory (Behrouz, Razaviyayn, et al. 2025; Sun, Li, et al. 2024; Wang et al. 2025). Accordingly, it is natural to ask how our discussions of memory capacity as well as Omega rule can affect Transformers. In this section, we discuss how our formulation of Omega rule is connected to Transformers and their sliding window counterparts (i.e., SWA). We then further provide two extensions to Transformers, each of which is a strict generalization of Transformers.

4.1 Online and Local Context Optimization of Memory

Connection to Sliding Window Attention. `Softmax` attention block can also be reformulated as a non-parametric solution to the $\ell_2(\cdot)$ regression with Nadaraya-Watson estimators (Fan 2018; Zhang et al. 2022):

$$\mathcal{M}^* = \arg\min_{\mathcal{M}} \sum_{i=1}^{L} \mathbf{s}(\mathbf{k}_i, \mathbf{q}) \|\mathbf{v}_i - \mathcal{M}\|_2^2 = \sum_{i=1}^{L} \frac{\mathbf{s}(\mathbf{k}_i, \mathbf{q})}{\sum_{j=1}^{L} \mathbf{s}(\mathbf{k}_j, \mathbf{q})} \mathbf{v}_i, \tag{17}$$

where L is the sequence length. While this formulation optimizes the memory $\mathcal{M}$ with respect to the entire sequence length, one can limit the optimization process to the past c tokens, resulting in:

$$\mathcal{M}^* = \arg\min_{\mathcal{M}} \sum_{i=t-c+1}^{t} \mathbf{s}(\mathbf{k}_i, \mathbf{q}_i) \|\mathbf{v}_i - \mathcal{M}\|_2^2 = \sum_{i=t-c+1}^{t} \frac{\mathbf{s}(\mathbf{k}_i, \mathbf{q})}{\sum_{j=t-c+1}^{t} \mathbf{s}(\mathbf{k}_j, \mathbf{q})} \mathbf{v}_i, \tag{18}$$

which is equivalent to the sliding window attention (SWA). This connection provides an important insight on the difference of attention and recurrent models: Not only attention is a non-parametric solution (contrary to the parametric nature of recurrent models), it globally optimizes its internal objective (attentional bias), while most recent modern recurrent models are online learners (Behrouz, Razaviyayn, et al. 2025; Peng, Zhang, et al. 2025; Sun, Li, et al. 2024; Yang, Kautz, et al. 2024)[1] . Our formulations of sliding window RNN and Omega rule fill this gap by optimizing the memory with respect to a context window of past tokens based on parametric methods, effectively memorizing the context instead of individual tokens.

Deep Linear Attention. As a novel baseline, we present Deep (Gated) Linear Attention (DLA) that replaces a matrix-valued memory in (gated) linear attention (Katharopoulos et al. 2020; Yang, Wang, Shen, et al. 2024) with a deep neural network (e.g., k-layer MLP). As discussed earlier in (Hebbian Rule), using dot product similarity as the internal attentional bias results in linear attention. Thus, leveraging recent deep memory modules (Behrouz, Razaviyayn, et al. 2025; Behrouz, Zhong, et al. 2024; Sun, Li, et al. 2024), we optimize the memory using gradient descent with dot product attentional bias:

$$\mathcal{M}_t = \alpha_t \mathcal{M}_{t-1} - \eta_t \nabla \ell(\mathcal{M}_{t-1}; \phi(\mathbf{k}_t), \mathbf{v}_t), \tag{19}$$

where $\ell(\mathcal{M}_{t-1}; \phi(\mathbf{k}_t), \mathbf{v}_t) = \langle \mathcal{M}_{t-1}(\phi(\mathbf{k}_t)), \mathbf{v}_t \rangle$ and $\phi(\cdot)$ is a polynomial kernel. The training of DLA can simply be parallelized using the hybrid of linear and non-linear chunk-wise training, the same as Behrouz, Zhong, et al. (2024) and Sun, Li, et al. (2024) and our discussion in Section 3.3.

Sliding Window Linear Attention. Building upon the above intuition and the connection of our formulation to SWA, we present Sliding Window Linear Attention (SWLA) block. Following the formulation of linear attention in associative memory perspective (Behrouz, Razaviyayn, et al. 2025), we use dot product similarity (i.e., $\ell(\mathcal{M}_t; \mathbf{k}_i, \mathbf{v}_i) = \langle \mathcal{M}_t(\mathbf{k}_i), \mathbf{v}_i \rangle$) as the attentional bias and optimize the loss function using gradient descent. For the sake of clarity, we use a linear memory here to derive the closed form:

$$\mathcal{M}_t = \alpha_t \mathcal{M}_{t-1} - \eta_t \nabla \sum_{i=t-c+1}^{t} \ell(\mathcal{M}_{t-1}; \phi(\mathbf{k}_i), \mathbf{v}_i) = \mathcal{M}_{t-1} + \sum_{i=t-c+1}^{t} \gamma_i^{(t)} \mathbf{v}_i \phi(\mathbf{k}_i)^\top \tag{20}$$

In the online case ($c = 1$) and $\phi(\cdot) = (\cdot)$, this recurrence is the same as linear attention (Katharopoulos et al. 2020).

4.2 Memory Capacity and Exponential Kernels

We first recall the formulation of `softmax` attention in Transformers (i.e., Equation 1):

$$\mathbf{y}_i = \frac{1}{\sum_{\ell=1}^{i} \exp\left(\mathbf{q}_i^\top \mathbf{k}_\ell / \sqrt{d_{\text{in}}}\right)} \sum_{j=1}^{i} \exp\left(\mathbf{q}_i^\top \mathbf{k}_j / \sqrt{d_{\text{in}}}\right) \mathbf{v}_j, \tag{21}$$

which its $\exp(\cdot)$ kernel is not separable and so cannot be written as a recurrence. Following the discussion in Kacham et al. (2024b), one can see $\exp(\cdot)$ kernel (compared to polynomial kernel $\phi_p(\cdot)$) as a feature map that maps the input into an

[1]Two of the exceptions are Titans (Behrouz, Zhong, et al. 2024) and Mesa-layer (Von Oswald et al. 2023), where Mesa-layer optimizes the memory with respect to all past tokens (comes with the cost of slow training), and Titans optimizes the memory with respect to all past tokens but with an implicit decay term (i.e., the result of the momentum) for each past token, maintaining parallelizability.

infinite dimension. That is, we define:

$$\phi^*(x) = \begin{pmatrix} 1 \\ \frac{x}{\sqrt{1}} \\ \frac{x^{\otimes 2}}{\sqrt{2!}} \\ \frac{x^{\otimes 3}}{\sqrt{3!}} \\ \vdots \end{pmatrix}, \qquad \phi_p(x) = x^{\otimes p}, \tag{22}$$

where $x^{\otimes p} = x \otimes x^{\otimes (p-1)}$ is a "self-tensoring" operator with Kronecker product (Kacham et al. 2024b) and so:

$$\exp(\mathbf{q}_t^\top \mathbf{k}_t) = \phi^*(\mathbf{q}_t)^\top \phi^*(\mathbf{k}_t). \tag{23}$$

Based on the above kernel, we can reformulate the attention (see Equation 21) as: (we remove $1/\sqrt{d_{\text{in}}}$ term for the sake of simplicity)

$$\mathbf{y}_i = \frac{1}{\sum_{\ell=1}^{i} \exp\left(\mathbf{q}_i^\top \mathbf{k}_\ell / \sqrt{d_{\text{in}}}\right)} \sum_{j=1}^{i} \mathbf{v}_j \phi^*(\mathbf{k}_j)^\top \phi^*(\mathbf{q}_i) = \frac{1}{\sum_{\ell=1}^{i} \exp\left(\mathbf{q}_i^\top \mathbf{k}_\ell / \sqrt{d_{\text{in}}}\right)} \left(\sum_{j=1}^{i} \phi^*(\mathbf{v}_j \mathbf{k}_j)^\top \right) \phi^*(\mathbf{q}_i) = \mathcal{M}_i \phi^*(\mathbf{q}_i), \tag{24}$$

This formulation, provides another important insight on the differences of attention and (kernel) recurrent models: `Softmax` attention as an associative memory has an unbounded memory and so can better memorize larger context into its parameters. Building upon this insight, we present DeepTransformers by replacing polynomial kernel with $\phi^*(\cdot)$ kernel in Deep Linear Attention formulation (Equation 19), resulting in unnormalized formulation of:

$$\mathcal{M}_t = \mathcal{M}_{t-1} - \nabla \langle \mathcal{M}_{t-1}(\phi^*(\mathbf{k}_t)), \mathbf{v}_t \rangle. \tag{25}$$

In the special case of linear memory, we can derive the closed form for the above formulation as:

$$\mathcal{M}_t = \mathcal{M}_{t-1} - \nabla \langle \mathcal{M}_{t-1} \phi^*(\mathbf{k}_t), \mathbf{v}_t \rangle = \mathcal{M}_{t-1} + \mathbf{v}_t \phi^*(\mathbf{k}_t)^\top = \sum_{i=1}^{t} \mathbf{v}_i \phi^*(\mathbf{k}_i)^\top \quad \Rightarrow \quad \mathbf{y}_t = \mathcal{M}_t \phi^*(\mathbf{q}_t) = \sum_{i=1}^{t} \mathbf{v}_i \exp(\mathbf{q}_i^\top \mathbf{k}_i), \tag{26}$$

which matches the output of the unnormalized Transformers. Therefore, DeepTransformers are strict generalizations of Transformers with `softmax` attention (Vaswani et al. 2017).

4.3 Deep Omega Transformer (Dot): Transformers with Omega learning rule

Our above formulation of DeepTransformers is based on the (Hebbian Rule), which is also used in original Transformers. However, as discussed earlier, using more powerful memory management and learning rules in associative memory modules can further enhance their performance. To this end, we extend the above formulation by replacing the Hebbian rule with our Omega learning rule, resulting in an unnormalized formulation of Deep Omega Transformers (Dot):

$$\mathcal{M}_t = \mathcal{M}_{t-1} - \nabla \sum_{i=t-c+1}^{t} \gamma_i^{(t)} \left\| \mathcal{M}\left(\phi^*(\mathbf{k}_i)\right) - \mathbf{v}_i \right\|_2^2. \tag{27}$$

We now discuss special instances of Dot to provide further intuition on its generalized formulation.

Linear Memory. This setup results in the following unnormalized formulation:

$$\mathcal{M}_t = \left(\mathbf{I} - \sum_{i=t-c+1}^{t} \gamma_i^{(t)} \phi^*(\mathbf{k}_i) \phi^*(\mathbf{k}_i)^\top \right) \mathcal{M}_{t-1} - \sum_{i=t-c+1}^{t} \gamma_i^{(t)} \mathbf{v}_i \phi^*(\mathbf{k}_i)^\top \tag{28}$$

$$\Rightarrow \mathbf{y}_t = \mathcal{M}_t \phi^*(\mathbf{q}_t) = \left(\mathbf{I} - \sum_{i=t-c+1}^{t} \gamma_i^{(t)} \phi^*(\mathbf{k}_i) \phi^*(\mathbf{k}_i)^\top \right) \mathcal{M}_{t-1} \phi^*(\mathbf{q}_t) - \sum_{i=t-c+1}^{t} \gamma_i^{(t)} \mathbf{v}_i \exp(\mathbf{q}_t^\top \mathbf{k}_i). \tag{29}$$

Online Case with $c = 1$. We now let $c = 1$:

$$\mathcal{M}_t = \left(\mathbf{I} - \eta_t \phi^*(\mathbf{k}_t)\phi^*(\mathbf{k}_t)^\top\right) \mathcal{M}_{t-1} - \eta_t \mathbf{v}_t \phi^*(\mathbf{k}_t)^\top \tag{30}$$

$$\Rightarrow \mathbf{y}_t = \mathcal{M}_t \phi^*(\mathbf{q}_t) = \left(\mathbf{I} - \eta_t \phi^*(\mathbf{k}_t)\exp(\mathbf{q}_t^\top \mathbf{k}_t)\right) \mathcal{M}_{t-1} - \eta_t \mathbf{v}_t \exp(\mathbf{q}_t^\top \mathbf{k}_t). \tag{31}$$

The above (unnormalized) formulation can be seen as the generalization of Transformers with Delta Rule. Therefore, due to the unbounded memory, Dot not only appends the new keys and values (similar to original Transformers), but it also replaces the new value with its predicted value from the previous state.

5 Atlas: A Locally Optimal Memory with High Capacity

Although the design of Omega rule allows the model to memorize the context instead of individual tokens and also the use of polynomial (or exponential) feature mapping increases memory capacity, the memory management (i.e., optimization of mappings between keys and values) is still limited to a simple gradient descent. This choice of optimizer can lead the model to a low-quality solution at a local optima, damaging the performance of the model in longer contexts. To overcome this issue, we suggest using Muon optimizer (Jordan et al. 2024) (with weight decay) that not only approximates second-order information, but it also mostly leverages matrix multiplication and can be parallelized across the sequence. Accordingly, the use of Muon for optimizing the internal objective in Equation 9, results in the following update rule:

$$\mathcal{M}_t = \alpha_t \mathcal{M}_{t-1} - \eta_t \ \texttt{NewtonShulz-}k(\mathcal{S}_t), \tag{32}$$

$$\mathcal{S}_t = \theta_t \mathcal{S}_{t-1} + \nabla \sum_{i=t-c+1}^{t} \gamma_i^{(t)} \left\| \mathcal{M}\left(\phi^*(\mathbf{k}_i)\right) - \mathbf{v}_i \right\|_2^2, \tag{33}$$

where c is the local context length and k is the number steps for `NewtonShulz` operations. For the additional discussion on the algorithm and this operation we refer the reader to Jordan et al. (2024). Following the literature on Muon optimizer, we know that when $k \to \infty$, then `NewtonShulz`-$k(\mathcal{S}_t)$ converges to the nearest semi-orthogonal matrix to the momentum term $\mathcal{S}_t$ and so approximate second-order information with a lower error. Therefore, interestingly, parameter k can be considered as an internal test-time compute parameter in Atlas, where using more steps can potentially result in better memorization.

5.1 Parallel Training

In this section, we discussed how the training process of Atlas can be parallelized. For the sake of clarity, we assume $c = 1$. Generalizing the process to arbitrary value for c follows the procedure in Section 3.3. We use the same process as we discussed in Section 3.3 and so chunk the sequence and compute all the gradients with respect to the last state of the previous chunk. Accordingly, using the recurrence of Atlas with momentum but without , we have:

$$\mathcal{M}_t = \alpha_t \mathcal{M}_{t-1} + \mathcal{S}_t \tag{34}$$

$$\mathcal{S}_t = \theta_t \mathcal{S}_{t-1} - \eta_t \nabla \ell(\mathcal{M}_{t'}; \mathbf{k}_t, \mathbf{v}_t). \tag{35}$$

Since t' is the last state of the previous chunk, we can calculate all the gradients before hand and so we let $u_t = \nabla \ell(\mathcal{M}_{t'}; \mathbf{k}_t, \mathbf{v}_t)$. Therefore, we have:

$$\mathcal{M}_t = \alpha_t \mathcal{M}_{t-1} + \mathcal{S}_t \tag{36}$$

$$\mathcal{S}_t = \theta_t \mathcal{S}_{t-1} - \eta_t u_t. \tag{37}$$

Now by expanding the second recurrence, we have:

$$\mathcal{S}_t = \theta_t \mathcal{S}_{t-1} - \eta_t \underbrace{\nabla \ell(\mathcal{M}_{t'}; \mathbf{k}_t, \mathbf{v}_t)}_{u_t}, \tag{38}$$

$$\Rightarrow \mathcal{S}_t = \underbrace{\theta_t \ldots \theta_1}_{\beta_t} \mathcal{S}_0 - \sum_{i=1}^{t} \frac{\theta_t \ldots \theta_1}{\theta_i \ldots \theta_1} \eta_i u_i = \beta_t \mathcal{S}_0 - \Theta \odot E \odot G, \tag{39}$$

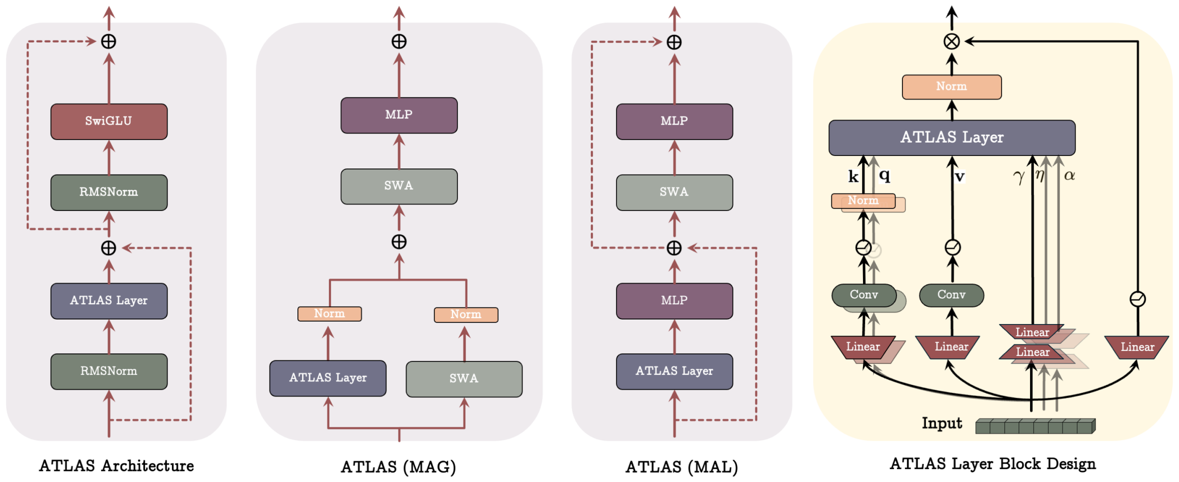


Figure 3: Visualization of the Atlas's (and our other variants') architecture, and its hybrid counterpart with SWA.

where G is the gradient matrix, E and Θ are diagonal matrices with value θ and η, and $\odot$ is broadcasting.

The main advantage of the above formulation (chunk wise recurrence) is that the recurrence of momentum is independent of the state of memory. That is, we can calculate all the momentum terms in the beginning of the chunk using the above formulation. Now in the Muon case, we want to use Newton-Schulz algorithm on the momentum terms, which results in:

$$\mathcal{S}'_t \leftarrow \texttt{Newton-Schulz5}(\mathcal{S}_t), \tag{40}$$

$$\mathcal{M}_t = \mathcal{M}_{t-1} + \mathcal{S}'_t. \tag{41}$$

Since the calculation of all $\mathcal{S}_t$s can be done in parallel, the calculation of `Newton-Schulz5`$(\cdot)$ can also be done in parallel.

Architectural Backbone. As for the architectural backbone, we follow the recent modern recurrent models (Allen-Zhu 2025; Arora, Eyuboglu, Zhang, et al. 2024; Behrouz, Zhong, et al. 2024; Yang, Wang, Zhang, et al. 2024) and use linear layers to project keys, values, and queries, followed by short convolution layers with size 4. We apply normalization on keys and queries to stabilize the training. We also follow Behrouz, Zhong, et al. (2024) and use two hybrid variants of MAL and MAG for our Atlas model. The architectures are illustrated in Figure 3. For models with deep memory architectures we use 2-layer MLP with residual connections:

$$\mathcal{M}(\cdot) = (\cdot) + W_1 \sigma(W_2(\cdot)). \tag{42}$$

We further extend this memory architecture, which is commonly used in recent studies (Behrouz, Razaviyayn, et al. 2025; Behrouz, Zhong, et al. 2024; Irie et al. 2021), to gated MLP layer as:

$$\mathcal{M}(\cdot) = (\cdot) + W_1 \left(\sigma\left(W_2(\cdot)\right) \otimes W_3(\cdot)\right), \tag{43}$$

where W_1, W_2, W_3 are linear learnable matrices. We refer to Atlas with the above memory architecture as Atlas++.

6 Experiments

Next, we evaluate the performance of Atlas, OmegaNet, DeepTransformers, and Dot in language modeling, commonsense reasoning, needle in haystack, and in-context recall tasks. Although we also discussed several other variants, such as SWLA, in our experiments we focus on the above models so in addition to comparison with state-of-the-art models, we also answer the following questions:

1. Is deep memory effective for `softmax` attention? (see Table 2 — comparison of Transformer++ and DeepTransformers)

2. Does the use of Omega improve the performance `softmax` attention? (see Table 2 — comparison of Transformer++, DeepTransformers, and Dot)
3. Does the Omega rule provide more expressive memory update? (see Table 2 and Table 6 — the performance of OmegaNet, and Atlas)
4. Is locally optimal memory update effective? (see Table 2 and Table 6 — comparison of OmegaNet, and Atlas)
5. Is non-linear feature mapping effective? (see Table 6)
6. Can the proposed improvements close the gap with Transformers in in-context recall tasks? (see Table 5)
7. What is the effect of the internal optimizer on the memory? (see Figure 6)

Setup. We train our models with training context window of size 4K using FineWeb dataset (Penedo et al. 2024). We use model size of 340M, 400M, 790M, and 1.3B parameters and train them on 15B, 15B, 30B, and 100B tokens sampled from the dataset. Baseline results are reported by Behrouz, Razaviyayn, et al. (2025), Behrouz, Zhong, et al. (2024), and Yang, Kautz, et al. (2024). Perplexity is measured on held-out validation data. As for the downstream tasks, we evaluate trained models on Wikitext (Merity et al. 2017), LMB (Paperno et al. 2016), PIQA (Bisk et al. 2020), HellaSwag (Zellers et al. 2019), WinoGrande (Sakaguchi et al. 2021), ARC-easy (ARC-e) and ARC-challenge (ARC-c) (Clark, Cowhey, et al. 2018), SIQA (Sap et al. 2019), and BoolQ (Clark, Lee, et al. 2019). Additional details about the experimental setups and other used datasets are in Appendix E.

6.1 Language Modeling and Common-Sense Reasoning

The results for Atlas, and OmegaNet as well as their corresponding baselines of SWDT, DLA, DeepTransformers, and Dot with the size of 760M and 1.3B are reported in Table 2. (see Appendix F for the results of small scale). Among non-hybrid models, including Transformer++, our Atlas, and OmegaNet achieve the best performance in both perplexity and accuracy measures. We attribute this performance to their ability to memorize the context rather than individual tokens. Comparing OmegaNet with Titans, that also uses the same momentary objective (i.e., ℓ_2 loss), but with context window of 1, we can observe the effectiveness of having non-online learning rule. On the other hand, our models, alone without any attention, can outperform hybrid models, while their hybrid variant of MAG further improve their performance. This performance gain is also related to the use of polynomial kernels that enhance the memory capacity of the model. See Table 6 for a more controlled study on the effect of different components.

Comparing Transformer++ with our more generalized Transformers (i.e., DeepTransformers, and Dot) we observe a consistent performance improvement. We attribute this performance to their deep memory, which makes them more powerful to model the dependencies of tokens. Comparing Dot with DeepTransformers, we can see the advantage of Omega rule, which helps the model to better manage its memory.

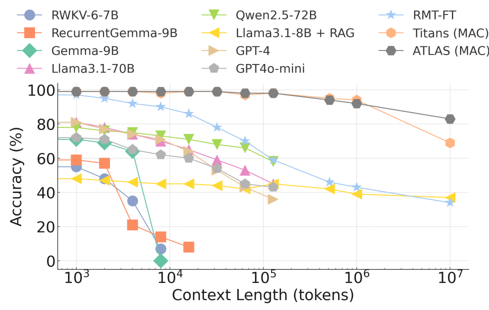


Figure 4: Performance of Atlas and baselines on BABILong benchmark. Atlas surpasses Titans performance and effectively scale to 10M context length in this task.

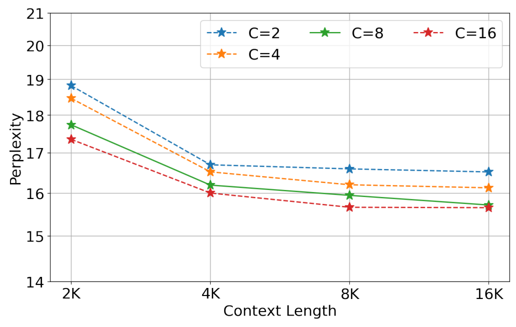


Figure 5: The effect of local context length (i.e. c) on the performance of OmegaNet with different global context length.

Table 2: Performance of ATLAS and baselines on language modeling and common-sense reasoning tasks. Hybrid models are marked with *. The best results are highlighted highlighted.

Model	**Wiki.** ppl ↓	**LMB.** ppl ↓	**LMB.** acc ↑	**PIQA** acc ↑	**Hella.** acc_n ↑	**Wino.** acc ↑	**ARC-e** acc ↑	**ARC-c** acc_n ↑	**SIQA** acc ↑	**BoolQ** acc ↑	**Avg.** ↑
					760M params / 30B tokens						
Transformer++	25.21	27.64	35.8	66.9	42.2	51.9	60.4	32.5	39.5	60.4	48.69
DEEPTRANSFORMERS (ours)	20.32	20.67	36.9	68.4	49.8	52.8	65.7	34.9	40.2	61.8	51.31
DOT (ours)	19.96	20.15	39.0	69.1	50.7	53.1	66.2	37.0	40.3	63.7	52.39
RetNet	26.08	24.45	34.5	67.2	41.6	52.1	63.2	32.8	38.4	57.9	48.46
DeltaNet	24.37	24.60	37.1	66.9	42.0	50.7	64.9	31.4	39.9	59.0	48.97
TTT	24.17	23.51	34.7	67.3	43.9	51.0	64.5	33.8	40.2	59.6	47.32
Gated DeltaNet	21.18	22.09	35.5	68.0	44.9	50.7	66.9	33.1	39.2	59.1	49.69
Samba*	20.63	22.71	39.7	69.2	47.4	52.0	66.9	33.2	39.0	61.2	51.08
Gated DeltaNet-H2*	19.88	20.83	39.2	69.0	48.2	52.6	67.0	35.5	39.4	61.1	51.49
Titans (LMM)	20.04	21.96	37.4	69.3	48.5	52.3	66.3	35.8	40.1	62.8	51.56
MEMORA	22.28	22.31	38.2	67.8	49.3	53.3	63.6	36.1	40.9	63.0	51.52
SWDT (ours)	19.89	21.52	36.2	68.3	45.2	53.0	65.4	34.2	39.5	59.5	50.1
DLA (ours)	23.12	22.09	36.1	68.0	47.9	52.7	65.8	34.6	39.1	59.6	50.46
OMEGANET (ours)	19.16	20.14	38.7	69.8	50.0	53.3	67.8	36.8	39.6	64.4	52.56
ATLAS (ours)	18.92	21.01	39.1	69.7	50.2	53.5	67.5	37.1	40.7	64.3	52.77
ATLAS++ (ours)	19.04	20.03	39.7	69.7	51.1	53.2	68.2	37.4	40.9	64.4	53.09
ATLAS (MAG)	18.62	21.18	40.0	70.3	50.5	53.0	68.1	36.5	41.2	65.0	53.08
ATLAS (MAL)	19.07	21.46	38.8	69.2	50.5	53.6	67.3	36.1	41.0	64.5	52.63
					1.3B params / 100B tokens						
Transformer++	18.53	18.32	42.6	70.0	50.2	53.5	68.8	35.1	40.7	57.1	52.25
DEEPTRANSFORMERS (ours)	15.67	12.63	49.4	72.6	57.0	58.8	71.1	37.5	41.6	61.5	56.19
DOT (ours)	15.28	11.96	50.1	73.3	57.5	60.4	72.2	41.2	42.7	61.4	57.35
RetNet	19.08	17.27	40.5	70.1	49.2	54.1	67.3	33.8	40.8	60.4	52.02
Mamba2	16.56	12.56	45.7	71.9	55.7	55.2	72.5	37.9	40.2	60.1	54.89
DeltaNet	17.71	16.88	42.5	70.7	50.9	53.3	68.5	35.7	40.2	55.3	52.14
Gated DeltaNet	16.42	12.17	46.6	72.2	55.8	57.4	71.2	38.4	40.6	60.2	55.32
Samba*	16.13	13.29	44.9	70.9	53.4	55.6	68.8	36.2	40.0	62.1	54.00
Gated DeltaNet-H2*	15.91	12.55	48.8	72.2	56.9	57.8	71.4	39.1	41.2	61.6	56.18
Titans (LMM)	15.60	11.41	49.1	73.1	56.3	59.8	72.4	40.8	42.1	61.0	56.82
MEMORA	15.90	12.04	48.7	73.1	56.0	57.4	71.5	37.9	40.2	61.3	55.87
OMEGANET (ours)	14.91	11.26	49.7	73.4	57.6	59.7	72.6	40.3	42.4	62.1	57.23
ATLAS (ours)	14.97	10.98	50.1	73.9	57.3	60.2	72.8	41.0	42.9	62.8	57.62
ATLAS++ (ours)	14.40	10.72	50.8	73.5	59.4	61.1	71.3	43.7	42.5	61.9	58.03

6.2 Long Context: Needle In a Haystack

One of our main motivations to design ATLAS is to enhance the performance of long-term neural memory module in long context tasks. Accordingly, to evaluate the effectiveness of our designs for improving the effective context length and memory capacity, we perform an experiment on needle-in-haystack tasks of RULER (Hsieh et al. 2024) benchmark. The performance of ATLAS and its hybrid variants, as well as our Transformer-like architectures and baselines are reported in Table 3. ATLAS shows very good performance compared to the recurrent baselines, outperforming modern recurrent neural networks such as Titans and DeltaNet. Its hybrid variants further improve its effective context length, effectively extrapolating to sequences with ×4 of their training context size. We attribute this performance to the proposed enhancements for the capacity of the memory. We further perform ablation studies to validate this claim. Also, our Transformer-like architectures outperforms the baselines, even our hybrid variants of ATLAS in longer contexts. This shows the importance of exponential feature mapping in longer sequences.

6.3 Long Context: BABILong Benchmark

To compare the effectiveness of ATLAS with Titans (Behrouz, Zhong, et al. 2024) in ultra-large sequences, we further evaluate ATLAS's performance on BABILong benchmark (Kuratov et al. 2024). In this experiment, we follow Behrouz,

Table 3: Performance of ATLAS and baselines on S-NIAH task from RULER benchmark. The best results among simple and hybrid models are highlighted.

Model	S-NIAH-PK				S-NIAH-N				S-NIAH-W		
	2K	4K	8K	16K	2K	4K	8K	16K	2K	4K	8K
TTT	98.4	98.8	98.0	88.4	60.2	36.6	10.2	4.4	78.8	28.0	4.4
DeltaNet	96.8	98.8	98.6	71.4	47.2	15.4	12.8	5.4	46.2	20.0	1.6
Titans (LMM)	99.8	98.4	98.2	96.2	100.0	99.8	93.4	80.2	90.4	89.4	85.8
ATLAS	100	99.2	98.0	97.0	100.0	100.0	93.0	84.0	93.2	90.6	86.2
Samba	98.8	98.0	97.4	97.2	98.8	98.6	96.2	95.6	96.8	90.0	84.0
Gated DeltaNet-H2*	99.2	97.8	97.4	98.4	98.0	97.8	96.2	95.8	97.4	96.8	88.4
ATLAS (MAG)	100	100	99.4	98.6	100	99.2	97.4	97.0	99.4	98.2	92.4
ATLAS (MAL)	99.8	99.6	98.4	96.8	99.8	98.0	97.2	96.8	98.0	98.4	92.6
DEEPTRANSFORMERS	100	100	98.2	97.8	100	98.8	97.8	94.0	95.8	92.2	88.4
DOT	100	100	99.6	98.6	100	100	97.8	96.8	99.0	98.4	93.2

Zhong, et al. (2024) and use MAC architecture but without persistent memory tokens. We also follow the original setup in the benchmark and fine-tune our model. The results are reported in Figure 4. While ATLAS shows competitive and on par performance with Titans until 1M context length, the performance of Titans drops in 10M. ATLAS, however, maintains its performance and achieve +80% accuracy in 10M context length. We attribute this to more powerful memory; in terms of (1) memory management (i.e., the use of Muon), (2) better memory capacity due to polynomial kernels, and (3) its nature to memorize the context, instead of individual tokens.

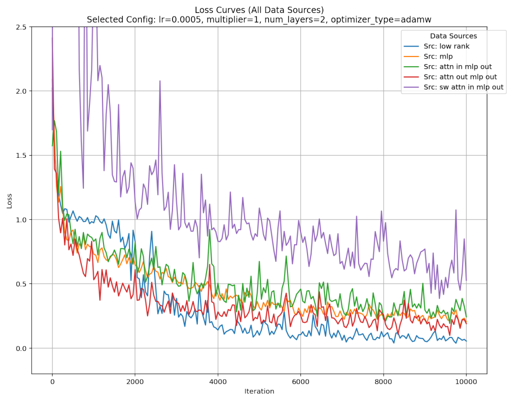
In previous sections, we show the effectiveness of our Transformer-like architectures (i.e., DEEPTRANSFORMERS and DOT) in both language modeling and long-context needle-in-haystack tasks. From now on, we focus on our recurrent architectures (i.e., ATLAS, and OMEGANET) to show the importance of presented improvements.

6.4 Learnability Experiments

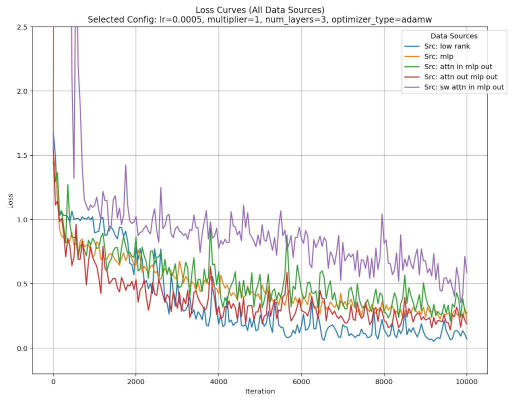
We have also performed some small-scale experiments to analyze the function-learning capability of small MLPs in an online fashion. In this setting, we have a sequence of tuples $(i_1, o_1), \ldots (i_t, o_t)$ with both $i_j, o_j \in \mathbb{R}^d$ for all j. We train an MLP $\mathcal{M}$ in an online fashion to minimize $\text{loss}_j = \|i_j - o_j\|_2^2 / \|o_j\|_2^2$ – specifically, we compute the gradient at time step j as $\nabla_{\mathcal{M}.\text{params}} \text{loss}_j$ and use standard optimizers such as Adam, Rmsprop and SGD to update the parameters. Such experiments help us understand the representation power of the models we use to represent memory and the power of optimization algorithms to quickly learn the underlying sequence mapping.

We study five different sequence to sequence functions:

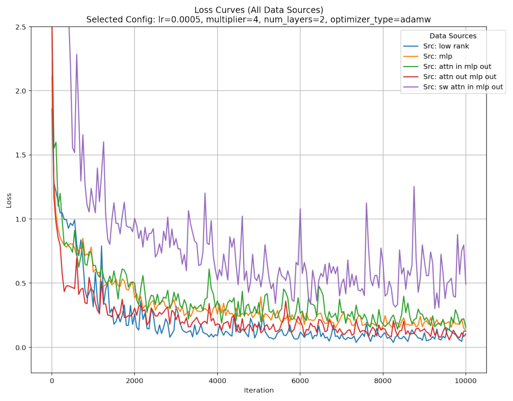
1. **Low Rank Mappings**: We sample a random low rank matrix $\mathbf{W} = \mathbf{X}\mathbf{Y}$ with $\mathbf{X} \in \mathbb{R}^{d \times k}$ and $\mathbf{Y} \in \mathbb{R}^{k \times d}$. We then sample $i_1, \ldots, i_t$ randomly from a Gaussian distribution and set $o_j = \mathbf{W}^\top \cdot i_j$ for all $j \in [t]$.
2. **MLP Mappings**: We sample an MLP $\mathcal{M}$ with 1 input, 1 hidden and 1 output layer which uses GELU non-linearity. We set the hidden dimension to d so that there is no expansion. We then sample $i_1, \ldots, i_t$ randomly from a Gaussian distribution and then set $o_j = \mathcal{M}(i_j)$ for all $j \in [t]$.
3. **Attention+MLP Mapping**: We sample $(i_1, \ldots, i_t)$ from a Gaussian distribution and an MLP $\mathcal{M}$ as above. We additionally sample three $d \times d$ matrices $\mathbf{W_Q}$, $\mathbf{W_K}$ and $\mathbf{W_V}$ and compute $q_j = \mathbf{W_Q}^\top \cdot i_j$, $k_j = \mathbf{W_K}^\top \cdot i_j$ and $v_j = \mathbf{W_K}^\top \cdot i_j$ for all $j \in [t]$. We then compute $o'_1, \ldots, o'_t$ as outputs of the causal masked attention mechanism applied on $\{q_j\}_{j \in [t]}, \{k_j\}_{j \in [t]}, \{v_j\}_{j \in [t]}$ and finally compute $o_j = \mathcal{M}(o_j)$.
4. **Attention Outputs as Inputs**: We do the same as above except that we output o'_j as the input sequence and o_j as the output sequence.
5. **Sliding Window Attention + MLP Mapping**: We do the same as in **Attention + MLP Mapping** setting except that we use a sliding window attention instead of full attention. We use a sliding window of 512 in our experiments.



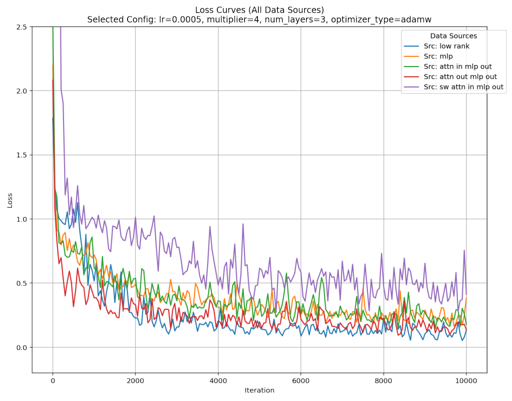
(a) $\mathcal{M}$ with 2 hidden layers and no expansion.



(b) $\mathcal{M}$ with 3 hidden layers and no expansion.



(c) $\mathcal{M}$ with 2 hidden layers and 4x expansion.



(d) $\mathcal{M}$ with 3 hidden layers and 4x expansion.

Figure 6: Loss curves for different setting with various hyperparameters

Table 4: Performance of Atlas, OmegaNet, and baselines on the synthetic benchmark of MAD (Poli et al. 2024). Atlas outperforms all the baselines, including Transformers.

	Compression	(Noisy) ICR	Fuzzy ICR	Selective Copying	Memorization	Average
Transformers	49.4	100	48.2	95.9	83.8	75.46
Gated DeltaNet	44.8	100	32.5	96.2	81.7	71.04
Titans	49.6	100	49.7	99.4	83.5	76.44
OmegaNet (ours)	50.9	100	54.2	99.6	90.2	78.98
Atlas (ours)	51.6	100	54.9	99.6	91.4	79.50

Note that the settings 3 and 5 are much harder to learn since they require (partially) memorizing the previous inputs and outputs to be able to learn the function that maps i_j to o_j, whereas the settings 1, 2 and 4 do not need to memorize the previous input-output pairs and just need to learn the underlying low-rank matrix or the MLP that maps the inputs to outputs.

The setting 4 is slightly different to setting 2 in that the inputs are not-independent at each time step and are correlated by the attention mechanism we use to compute the inputs. Thus a strong learning algorithm maybe able to utilize the underlying correlations to learn the mapping faster in setting 4 versus setting 2.

Table 5: The performance of our models (Atlas, and OmegaNet) compared to baselines. While still Transformers achieve the best results in in-context recall tasks, our design of context memorization and polynomial feature maps can close the gap with Transformers.

	SWDE	NQ	DROP	FDA	SQUAD	TQA	Average
Transformers	84.9	23.0	28.4	72.5	48.1	64.4	53.55
Gated DeltaNet	63.2	19.1	26.7	33.4	39.6	59.7	40.28
Titans	65.1	20.7	27.2	37.3	42.6	61.0	42.31
OmegaNet (ours)	<u>67.4</u>	21.1	27.2	39.0	43.2	60.9	43.13
Atlas (ours)	66.8	<u>21.9</u>	<u>27.4</u>	<u>40.7</u>	<u>44.1</u>	<u>61.3</u>	<u>43.70</u>

Table 6: Ablation Study on Atlas. All components of Atlas are positively contributing to its performance.

Model	Language Modeling ppl ↓	C.S. Reasoning acc ↑
Atlas	19.97	52.77
+Gated MLP Memory	19.53	53.09
+`Attn` (MAG)	19.90	53.08
+`Attn` (MAL)	20.26	52.63
Linear Memory	21.03	49.74
w/o Muon	19.65	52.56
$c = 1$	21.98	49.26
w/o Polynomial Mapping	22.14	50.57

We set $d = 256$ and show the loss curves vs sequence position for all the five settings with function learning MLP $\mathcal{M}$ being defined and trained with different settings in Figure 6. We can see that in all the settings, the model learns non-trivial mappings from inputs to outputs with the $loss_j = \|i_j - o_j\|_2^2 / \|o_j\|_2^2$ being smaller than 1 eventually. Most notably, the correlations in inputs in setting 4 induced by the attention mechanism makes the model quickly learn the mapping compared to in setting 2 and the models usually learn the best in setting 1 which is the least complex function.

The models do the worst in settings 3 and 5 which require the models to (partially) memorize the inputs and outputs to learn the attention mechanism outputs. Surprisingly, the models learn to do better in setting 3 vs setting 5, when we would expect that capacity requirement for setting 3 to be higher than setting 5. We hypothesize that the learning algorithm is unable to make the model 'forget' old inputs which makes the loss worse in sliding window setting when compared to global attention setting. A caveat of our analysis is that, the attention computation is done on randomly initialized vectors and hence the attention matrix is usually not spiky, unlike in the attention matrix for trained set of query, key and value vectors in LLMs. This leads to attention outputs being close to the mean of value vectors in the context.

6.5 Additional Experiments: In-context Recall, MAD Synthetic Benchmark, and Associative Recall

In this section, we first evaluate the performance of our models on MAD benchmark, a synthetic benchmark that evaluate the performance of models in recall, memorization, compression, and copying tasks (Poli et al. 2024). The results are reported in Table 4. Atlas achieves the best results in all aspects, particularly in memorization, which shows the importance of its components for enhancing the memory capacity.

In-context recall tasks is one of the most challenging benchmarks for recurrent neural networks. In this section, we follow Arora, Eyuboglu, Zhang, et al. (2024) and perform experiments on SWDE (Lockard et al. 2019), NQ (Kwiatkowski et al. 2019), DROP (Dua et al. 2019), FDA (Arora, Yang, et al. 2023), SQUAD (Rajpurkar et al. 2016), and TQA (Kembhavi et al. 2017) to evaluate and compare the performance of Atlas with baselines and Transformers. The results are reported in Table 5. While Transformers still achieve the best results in in-context recall tasks, Atlas and OmegaNet shows competitive performance and performs better than state-of-the-art recurrent models. We again attribute this performance to better memory management and capacity.

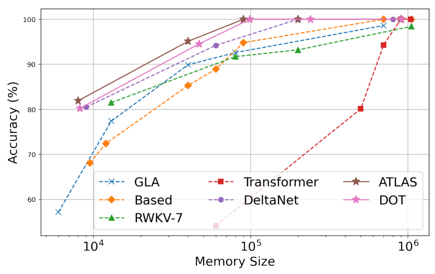


Figure 7: The results for associative memory recall.

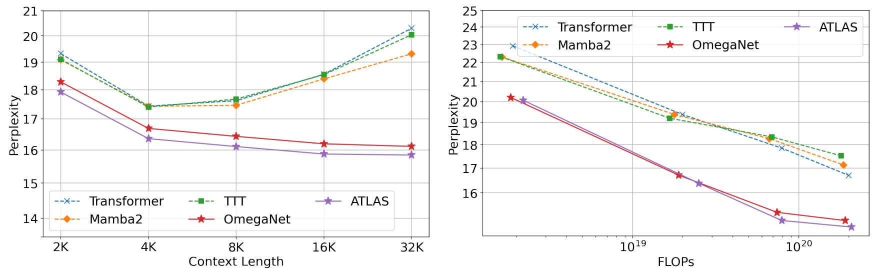


Figure 8: Scaling patterns of Atlas, and OmegaNet with respect to (Left) training context length, and (Right) FLOPs.

Finally, following Yang, Wang, Zhang, et al. (2024) and Arora, Eyuboglu, Timalsina, et al. (2023) we evaluate the performance of Atlas and Dot in Multi-Query Associative Recall (MQAR) task (Arora, Eyuboglu, Timalsina, et al. 2023). The results are reported in Figure 7. Both models show good performance compared to baselines and Atlas achieve the best performance per memory size compared to state-of-the-art models such as DeltaNet (Yang, Wang, Zhang, et al. 2024).

6.6 Ablation Study and Scaling Patterns

In this section, we perform an ablation study on the differernt components of Atlas, and also evaluate its scaling patterns with respect to the number of parameters and also the context length of the training. The results for ablation study are reported in Table 6. The results show that: (1) more powerful memory architectures such as gated MLP can further enhance the performance of Atlas; (2) The hybrid variants further improve the performance, where MAG shows better improvement compared to MAL architecture; (3) Polynomial mappings as well as deep memory are particularly important when we use context memorization (i.e., Omega rule). Figure 5 also shows the effect of local context length (i.e., c) on the performance of the model. With the increase of c we can achieve better performance, mainly due to the gating parameters of γ that can prune the context, whenever it is needed.

Model Size. Figure 8 shows the scaling pattern of Atlas, and OmegaNet, with respect to number of parameters and compared to baseline. Both models achieve a good scaling pattern with increasing the model size, achieving lower perplexity in all scales compared to baselines.

Context Length. Figure 8 shows the scaling pattern of Atlas, and OmegaNet, with respect to the context length and compared to baseline. Both models due to high memory capacity can scale well, when increasing the context length.

7 Conclusion

We introduced Atlas, a new long-term memory module designed to address the core limitations of modern recurrent models in long-context understanding: limited memory capacity, online-only updates, and weak memory management. Our proposed sliding window learning rule, higher-order feature mappings, and advanced memory optimizers offer a principled and scalable approach to overcoming these challenges. Empirically, our models—OmegaNet, Atlas, DeepTransformers, and Dot—achieve consistent improvements over Transformers and recent RNN variants across diverse benchmarks. Theoretically, we provided insight into memory capacity and optimization dynamics, offering explanations for the context length limitations observed in prior works.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. "Gpt-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023).

[2] Zeyuan Allen-Zhu. "Physics of Language Models: Part 4.1, Architecture Design and the Magic of Canon Layers". In: *SSRN Electronic Journal* (May 2025). https://ssrn.com/abstract=5240330.

[3] Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. "Zoology: Measuring and improving recall in efficient language models". In: *arXiv preprint arXiv:2312.04927* (2023).

[4] Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. "Simple linear attention language models balance the recall-throughput tradeoff". In: *Forty-first International Conference on Machine Learning*. 2024. URL: https://openreview.net/forum?id=e93ffDcpH3.

[5] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. "Language models enable simple systems for generating structured views of heterogeneous data lakes". In: *arXiv preprint arXiv:2304.09433* (2023).

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[7] Eric B Baum. "On the capabilities of multilayer perceptrons". In: *Journal of Complexity* 4.3 (1988), pp. 193–215. ISSN: 0885-064X. DOI: https://doi.org/10.1016/0885-064X(88)90020-9. URL: https://www.sciencedirect.com/science/article/pii/0885064X88900209.

[8] Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. "It's All Connected: A Journey Through Test-Time Memorization, Attentional Bias, Retention, and Online Optimization". In: *arXiv preprint arXiv:2504.13173* (2025).

[9] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. "Titans: Learning to memorize at test time". In: *arXiv preprint arXiv:2501.00663* (2024).

[10] Srinadh Bhojanapalli, Chulhee Yun, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. "Low-rank bottleneck in multi-head attention models". In: *International conference on machine learning*. PMLR. 2020, pp. 864–873.

[11] Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. "Birth of a transformer: A memory viewpoint". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 1560–1588.

[12] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. "Piqa: Reasoning about physical commonsense in natural language". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 2020, pp. 7432–7439.

[13] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. "BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 2924–2936. DOI: 10.18653/v1/N19-1300. URL: https://aclanthology.org/N19-1300/.

[14] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. "Think you have solved question answering? try arc, the ai2 reasoning challenge". In: *arXiv preprint arXiv:1803.05457* (2018).

[15] Thomas M. Cover. "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition". In: *IEEE Transactions on Electronic Computers* EC-14.3 (1965), pp. 326–334. DOI: 10.1109/PGEC.1965.264137.

[16] Róbert Csordás, Christopher Potts, Christopher D Manning, and Atticus Geiger. "Recurrent Neural Networks Learn to Store and Generate Sequences using Non-Linear Representations". In: *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*. 2024, pp. 248–262.

[17] Karan Dalal, Daniel Koceja, Gashon Hussein, Jiarui Xu, Yue Zhao, Youjin Song, Shihao Han, Ka Chun Cheung, Jan Kautz, Carlos Guestrin, et al. "One-Minute Video Generation with Test-Time Training". In: *arXiv preprint arXiv:2504.05298* (2025).

[18] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. "DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs". In: *arXiv preprint arXiv:1903.00161* (2019).

[19] Jianqing Fan. *Local polynomial modelling and its applications: monographs on statistics and applied probability 66*. Routledge, 2018.

[20] Xavier Gonzalez, Andrew Warrington, Jimmy Smith, and Scott Linderman. "Towards scalable and stable parallelization of nonlinear rnns". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 5817–5849.

[21] Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. "Liquid Structural State-Space Models". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=g4OTKRKfS7R.

[22] Zexue He, Leonid Karlinsky, Donghyun Kim, Julian McAuley, Dmitry Krotov, and Rogerio Feris. "CAMELoT: Towards Large Language Models with Training-Free Consolidated Associative Memory". In: *arXiv preprint arXiv:2402.13449* (2024).

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

[24] Dan Hendrycks and Kevin Gimpel. "Gaussian error linear units (gelus)". In: *arXiv preprint arXiv:1606.08415* (2016).

[25] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[26] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. "RULER: What's the Real Context Size of Your Long-Context Language Models?" In: *First Conference on Language Modeling*. 2024. URL: https://openreview.net/forum?id=kIoBbc76Sy.

[27] Jerry Yao-Chieh Hu, Dennis Wu, and Han Liu. "Provably optimal memory capacity for modern hopfield models: Transformer-compatible dense associative memories as spherical codes". In: *arXiv preprint arXiv:2410.23126* (2024).

[28] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. "Transformer quality in linear time". In: *International conference on machine learning*. PMLR. 2022, pp. 9099–9117.

[29] Guang-Bin Huang. "Learning capability and storage capacity of two-hidden-layer feedforward networks". In: *IEEE Transactions on Neural Networks* 14.2 (2003), pp. 274–281. DOI: 10.1109/TNN.2003.809401.

[30] Kazuki Irie, Imanol Schlag, Robert Csordas, and Jurgen Schmidhuber. "Going beyond linear transformers with recurrent fast weight programmers". In: *Advances in neural information processing systems* 34 (2021), pp. 7703–7717.

[31] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. *Muon: An optimizer for hidden layers in neural networks*. 2024. URL: https://kellerjordan.github.io/posts/muon/.

[32] Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. "PolySketchFormer: Fast Transformers via Sketching Polynomial Kernels". In: *Forty-first International Conference on Machine Learning*. 2024. URL: https://openreview.net/forum?id=ghYrfdJfjK.

[33] Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. "PolySketchFormer: Fast Transformers via Sketching Polynomial Kernels". In: *Proceedings of the 41st International Conference on Machine Learning*. Ed. by Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp. Vol. 235. Proceedings of Machine Learning Research. PMLR, July 2024, pp. 22748–22770. URL: https://proceedings.mlr.press/v235/kacham24a.html.

[34] Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. "Gemma 3 technical report". In: *arXiv preprint arXiv:2503.19786* (2025).

[35] M. Karami and V. Mirrokni. *Lattice: Learning to Efficiently Compress the Memory*. 2025.

[36] Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. "Finetuning Pretrained Transformers into RNNs". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 10630–10643. DOI: 10.18653/v1/2021.emnlp-main.830. URL: https://aclanthology.org/2021.emnlp-main.830/.

[37] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. "Transformers are rnns: Fast autoregressive transformers with linear attention". In: *International conference on machine learning*. PMLR. 2020, pp. 5156–5165.

[38] Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. "Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition*. 2017, pp. 4999–5007.

[39] Dmitry Krotov. "Hierarchical associative memory". In: *arXiv preprint arXiv:2107.06446* (2021).

[40] Dmitry Krotov and John J Hopfield. "Dense associative memory for pattern recognition". In: *Advances in neural information processing systems* 29 (2016).

[41] Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Igorevich Sorokin, Artyom Sorokin, and Mikhail Burtsev. "BABILong: Testing the Limits of LLMs with Long Context Reasoning-in-a-Haystack". In: *The Thirty-*

eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track. 2024. URL: https://openreview.net/forum?id=u7m2CG84BQ.

[42] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. "Natural questions: a benchmark for question answering research". In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 453–466.

[43] Chengxuan Li, Di Huang, Zeyu Lu, Yang Xiao, Qingqi Pei, and Lei Bai. "A survey on long video generation: Challenges, methods, and prospects". In: *arXiv preprint arXiv:2403.16407* (2024).

[44] Xiaoyu Li, Yuanpeng Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. "On the expressive power of modern hopfield networks". In: *arXiv preprint arXiv:2412.05562* (2024).

[45] Yi Heng Lim, Qi Zhu, Joshua Selfridge, and Muhammad Firmansyah Kasim. "Parallelizing non-linear sequential models over the sequence length". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=E34AlVLN0v.

[46] Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. "Longhorn: State space models are amortized online learners". In: *arXiv preprint arXiv:2407.14207* (2024).

[47] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. "Lost in the middle: How language models use long contexts". In: *Transactions of the Association for Computational Linguistics* 12 (2024), pp. 157–173.

[48] Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. "Openceres: When open information extraction meets the semi-structured web". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 3047–3056.

[49] Carlo Lucibello and Marc Mézard. "Exponential capacity of dense associative memories". In: *Physical Review Letters* 132.7 (2024), p. 077301.

[50] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. "Pointer Sentinel Mixture Models". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=Byj72udxe.

[51] William Merrill, Jackson Petty, and Ashish Sabharwal. "The Illusion of State in State-Space Models". In: *Forty-first International Conference on Machine Learning*. 2024. URL: https://openreview.net/forum?id=QZgo9JZpLq.

[52] Guido Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. "On the number of linear regions of deep neural networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, pp. 2924–2932.

[53] Tsendsuren Munkhdalai, Alessandro Sordoni, Tong Wang, and Adam Trischler. "Metalearned neural memory". In: *Advances in Neural Information Processing Systems* 32 (2019).

[54] Tsendsuren Munkhdalai and Hong Yu. "Neural semantic encoders". In: *Proceedings of the conference. Association for Computational Linguistics. Meeting*. Vol. 1. NIH Public Access. 2017, p. 397.

[55] Denis Paperno, German Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. "The LAMBADA dataset: Word prediction requiring a broad discourse context". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Katrin Erk and Noah A. Smith. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1525–1534. DOI: 10.18653/v1/P16-1144. URL: https://aclanthology.org/P16-1144/.

[56] Razvan Pascanu, Guido Montufar, and Yoshua Bengio. *On the number of response regions of deep feed forward networks with piece-wise linear activations*. 2014. arXiv: 1312.6098 [cs.LG]. URL: https://arxiv.org/abs/1312.6098.

[57] Guilherme Penedo, Hynek Kydlicek, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. "The fineweb datasets: Decanting the web for the finest text data at scale". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 30811–30849.

[58] Bo Peng, Eric Alcaide, Quentin Gregory Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Nguyen Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan S. Wind, Stanisław Wozniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. "RWKV: Reinventing RNNs for the Transformer Era". In: *The 2023 Conference on Empirical Methods in Natural Language Processing*. 2023. URL: https://openreview.net/forum?id=7SaXczaBpG.

[59] Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, et al. "Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence". In: *arXiv preprint arXiv:2404.05892* (2024).

[60] Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Haowen Hou, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, et al. “Rwkv-7" goose" with expressive dynamic state evolution”. In: *arXiv preprint arXiv:2503.14456* (2025).
[61] Michael Poli, Armin W Thomas, Eric Nguyen, Pragaash Ponnusamy, Bjorn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Re, et al. “Mechanistic design and scaling of hybrid architectures”. In: *arXiv preprint arXiv:2403.17844* (2024).
[62] DL Prados and SC Kak. “Neural network capacity using delta rule”. In: *Electronics Letters* 25.3 (1989), pp. 197–199.
[63] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. “Squad: 100,000+ questions for machine comprehension of text”. In: *arXiv preprint arXiv:1606.05250* (2016).
[64] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David Kreil, Michael K Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. “Hopfield Networks is All You Need”. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=tL89RnzIiCd.
[65] Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. “Samba: Simple Hybrid State Space Models for Efficient Unlimited Context Language Modeling”. In: *arXiv preprint arXiv:2406.07522* (2024).
[66] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. “Winogrande: An adversarial winograd schema challenge at scale”. In: *Communications of the ACM* 64.9 (2021), pp. 99–106.
[67] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. “Social IQa: Commonsense Reasoning about Social Interactions”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4463–4473. DOI: 10.18653/v1/D19-1454. URL: https://aclanthology.org/D19-1454/.
[68] Siddhartha Satpathi and Rayadurgam Srikant. “The dynamics of gradient descent for overparametrized neural networks”. In: *Learning for Dynamics and Control*. PMLR. 2021, pp. 373–384.
[69] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. “Linear transformers are secretly fast weight programmers”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9355–9366.
[70] JH Schmidhuber. “Learning to control fast-weight memories: An alternative to recurrent nets. Accepted for publication in”. In: *Neural Computation* (1992).
[71] Jürgen Schmidhuber. “Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets”. In: *ICANN’93: Proceedings of the International Conference on Artificial Neural Networks Amsterdam, The Netherlands 13–16 September 1993 3*. Springer. 1993, pp. 460–463.
[72] Jürgen Schmidhuber and Sepp Hochreiter. “Long Short-term Memory”. In: *Neural Computation MIT-Press* (1997).
[73] Mark Schöne, Babak Rahmani, Heiner Kremer, Fabian Falck, Hitesh Ballani, and Jannes Gladrow. “Implicit Language Models are RNNs: Balancing Parallelization and Expressivity”. In: *arXiv preprint arXiv:2502.07827* (2025).
[74] Jack Sherman and Winifred J Morrison. “Adjustment of an inverse matrix corresponding to a change in one element of a given matrix”. In: *The Annals of Mathematical Statistics* 21.1 (1950), pp. 124–127.
[75] Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. “DeltaProduct: Increasing the Expressivity of DeltaNet Through Products of Householders”. In: *arXiv preprint arXiv:2502.10297* (2025).
[76] Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. “Simplified State Space Layers for Sequence Modeling”. In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=Ai8Hw3AXqks.
[77] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. “Learning to (learn at test time): Rnns with expressive hidden states”. In: *arXiv preprint arXiv:2407.04620* (2024).
[78] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. “Retentive network: A successor to transformer for large language models”. In: *arXiv preprint arXiv:2307.08621* (2023).
[79] W Scott Terry. *Learning and memory: Basic principles, processes, and procedures*. Routledge, 2017.
[80] Matteo Tiezzi, Michele Casoni, Alessandro Betti, Tommaso Guidi, Marco Gori, and Stefano Melacci. “On the resurgence of recurrent models for long sequences: Survey and research opportunities in the transformer era”. In: *arXiv preprint arXiv:2402.08132* (2024).
[81] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL:

https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[82] Johannes Von Oswald, Maximilian Schlegel, Alexander Meulemans, Seijin Kobayashi, Eyvind Niklasson, Nicolas Zucchet, Nino Scherrer, Nolan Miller, Mark Sandler, Max Vladymyrov, et al. "Uncovering mesa-optimization algorithms in transformers". In: *arXiv preprint arXiv:2309.05858* (2023).

[83] Ke Alexander Wang, Jiaxin Shi, and Emily B Fox. "Test-time regression: a unifying framework for designing sequence models with associative memory". In: *arXiv preprint arXiv:2501.12352* (2025).

[84] Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. "Rnns are not transformers (yet): The key bottleneck on in-context retrieval". In: *arXiv preprint arXiv:2402.18510* (2024).

[85] Bernard Widrow and Marcian E Hoff. *Adaptive switching circuits*. 1988.

[86] David J Willshaw, O Peter Buneman, and Hugh Christopher Longuet-Higgins. "Non-holographic associative memory". In: *Nature* 222.5197 (1969), pp. 960–962.

[87] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. "Gated Delta Networks: Improving Mamba2 with Delta Rule". In: *arXiv preprint arXiv:2412.06464* (2024).

[88] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. "Gated Linear Attention Transformers with Hardware-Efficient Training". In: *Forty-first International Conference on Machine Learning*. 2024. URL: https://openreview.net/forum?id=ia5XvxFUJT.

[89] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. "Parallelizing linear transformers with the delta rule over sequence length". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 115491–115522.

[90] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. "HellaSwag: Can a Machine Really Finish Your Sentence?" In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Korhonen, David Traum, and Lluis Marquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 4791–4800. DOI: 10.18653/v1/P19-1472. URL: https://aclanthology.org/P19-1472/.

[91] Yufeng Zhang, Boyi Liu, Qi Cai, Lingxiao Wang, and Zhaoran Wang. "An analysis of attention via the lens of exchangeability and latent variable models". In: *arXiv preprint arXiv:2212.14852* (2022).

A Additional Related Work

Modern Linear Recurrent Neural Networks[2]. Recent research endeavors have concentrated on mitigating the quadratic computational complexity and inherent limitations of Transformer models in processing long-context sequences. This has led to the development of efficient recurrent alternatives, primarily motivated by their rapid inference and training capabilities (Tiezzi et al. 2024). Initial advancements in this domain, exemplified by models such as RetNet (Sun, Dong, et al. 2023), RWKV (Peng, Alcaide, et al. 2023), and S5 (Smith et al. 2023), employed data-independent transition matrices coupled with Hebbian-like update mechanisms. Subsequently, a second generation of models emerged, incorporating input-dependent parameters within these linear architectures (e.g., linear RNNs (Hasani et al. 2023; Smith et al. 2023), RWKV6 (Peng, Goldstein, et al. 2024)). These models also explored more expressive memory updating rules, notably those based on the delta rule (Liu, Wang, et al. 2024; Peng, Zhang, et al. 2025; Schlag et al. 2021; Yang, Kautz, et al. 2024; Yang, Wang, Zhang, et al. 2024). Further evolution in this line of research has extended these memory architectures to deeper models, while concurrently utilizing delta-rule-like update mechanisms (Sun, Li, et al. 2024) or data-dependent momentum-based update rules with forget gating (Behrouz, Zhong, et al. 2024). More recently, to augment the performance of delta-rule-based sequential models, Siems et al. (2025) have proposed the application of multiple gradient descent updates per token, thereby yielding more expressive sequence models, particularly in state tracking tasks. In addition to the above fast linear recurrent sequence models, several studies have focused on RNNs with non-linear recurrence (Behrouz, Razaviyayn, et al. 2025; Csordás et al. 2024; Gonzalez et al. 2024; Karami et al. 2025; Lim et al. 2024; Merrill et al. 2024; Schöne et al. 2025; Von Oswald et al. 2023), and how their training can be faster (Gonzalez et al. 2024; Lim et al. 2024; Schöne et al. 2025).

Fast Weight Programs. The conceptualization of linear layers as key-value associative memory systems can be traced back to Hopfield networks (Hopfield 1982). This concept was subsequently developed in the context of fast weight programmers, wherein dynamic fast programs are integrated into recurrent neural networks to serve as writable memory stores (Schlag et al. 2021; Schmidhuber 1992; Schmidhuber 1993). Among the learning paradigms for such systems, Hebbian learning (Hebb 2005) and the delta rule (Prados et al. 1989) have emerged as the most prominent. Both learning rules have been the subject of extensive investigation within the existing literature (Irie et al. 2021; Munkhdalai, Sordoni, et al. 2019; Munkhdalai and Yu 2017; Schlag et al. 2021; Schmidhuber 1992; Yang, Kautz, et al. 2024; Yang, Wang, Zhang, et al. 2024).

Hopfield Networks. Our formulation is architecturally founded upon the broad concept of associative memory, wherein the primary objective is to learn an underlying mapping between keys and values. Seminal work by Hopfield (1982) on Hopfield Networks introduced one of the earliest neural architectures explicitly based on associative memory, defining it through the minimization of an energy function for storing key-value pairs. Although traditional Hopfield networks have seen diminished applicability in recent years, primarily due to constraints in vector-valued memory capacity and the nature of their energy function, several contemporary studies have focused on enhancing their capacity through various methodologies. These include efforts by Krotov (2021), Li, Li, et al. (2024), and Krotov and Hopfield (2016). Notably, extensions to the energy function of these models, often incorporating exponential kernels, have been explored (Krotov and Hopfield 2016; Lucibello et al. 2024). Furthermore, the relationship between these modernized Hopfield networks and Transformer architectures has been a subject of recent investigation (Hu et al. 2024; Ramsauer et al. 2021).

B Miras Framework

As discussed earlier, Behrouz, Razaviyayn, et al. (2025) formalized the concept of associative memory as:

Definition 2 (Behrouz, Razaviyayn, et al. (2025))**.** *Given a set of keys $\mathcal{K} \subseteq \mathbb{R}^{d_k}$ and values $\mathcal{V} \subseteq \mathbb{R}^{d_v}$, associative memory is an mapping $\mathcal{M} : \mathcal{K} \to \mathcal{V}$. Learning the associative memory is based on an objective $\mathcal{L}$, called* Attentional Bias*, that determines the type of memory and its priorities:*

$$\mathcal{M}^* = \arg\min_{\mathcal{M}} \quad \mathcal{L}(\mathcal{M}(\mathcal{K}); \mathcal{V}). \tag{44}$$

[2]Note that here the term "linear" refers to their fast training and inference procedures. This does not refer to their recurrence formula as some models like Titans (Behrouz, Zhong, et al. 2024), Yaad, Moneta, Memora (Behrouz, Razaviyayn, et al. 2025), and TTT (Sun, Li, et al. 2024) are based on *non-linear* recurrence but fast at training and inference.

Optimizing this objective using an iterative algorithm (e.g., gradient descent) results in the memory update rule. Thus, the sequence model is a meta in-context learner with two optimization levels:

1. Inner Loop: Where parameters of the memory module are optimized (i.e., $\boldsymbol{\theta}_{\mathcal{M}} = \{W_1, W_2, \dots, W_{\mathcal{L}_{\mathcal{M}}, \dots}\}$). In the inner optimization loop, all other parameters from the model are considered hyperparameters and are fixed and *not* optimized.
2. Outer Loop: Where all other parameters of the model are optimized, such as linear projections, MLP layers, convolutions, etc.

B.1 Examples

As an example, one can define the linear attention as the optimization of dot-product similarity with gradient descent: i.e., $\tilde{\ell}_t := \langle \mathcal{M}_{t-1}\mathbf{k}_t, \mathbf{v}_t \rangle$. That is,

$$\mathcal{M}_t = \mathcal{M}_{t-1} - \eta_t \nabla \tilde{\ell}_t(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t) = \mathcal{M}_{t-1} - \eta_t \nabla \langle \mathcal{M}_{t-1}\mathbf{k}_t, \mathbf{v}_t \rangle \tag{45}$$

$$= \mathcal{M}_{t-1} + \eta_t \mathbf{v}_t \mathbf{k}_t^\top. \tag{46}$$

As an another example, if we use regression loss, instead of the dot-product similarity, we can obtain the DeltaNet (Schlag et al. 2021):

$$\mathcal{M}_t = \mathcal{M}_{t-1} - \eta_t \nabla \|\mathcal{M}_t \mathbf{k}_t - \mathbf{v}_t\|_2^2 = \mathbf{I} - \eta_t \mathbf{k}_t \mathbf{k}_t^\top \mathcal{M}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top. \tag{47}$$

C Supporting Proofs

Proposition 1 (Capacity of ℓ_2 Attentional Bias)**.** *Let $\mathcal{M}$ be a matrix-valued memory with $d_v \times d_k$ parameters that optimizes the internal objective of $\ell(\mathcal{M}_t; \mathbf{k}_t, \mathbf{v}_t) = \|\mathcal{M}_t \mathbf{k}_t - \mathbf{v}_t\|_2^2$ with gradient descent. $\mathcal{M}$ can store the mapping of at most $O(d_k)$ pairs of $(\mathbf{k}_i, \mathbf{v}_i)$ with linearly independent keys.*

Proof. Let $K = [\mathbf{k}_1 \cdots \mathbf{k}_m] \in \mathbb{R}^{d_k \times m}$ and $V = [\mathbf{v}_1 \cdots \mathbf{v}_m] \in \mathbb{R}^{d_v \times m}$. The optimization problem becomes minimizing the Frobenius norm $\|\mathcal{M}K - V\|_2^2$. Exact memorization requires solving the linear system $\mathcal{M}K = V$.

Vectorizing the expression yields the system $(K^\top \otimes I_{d_v})\text{vec}(\mathcal{M}) = \text{vec}(V)$, which has md_v scalar equations in $d_k d_v$ unknowns. When the keys are linearly independent, $\text{rank}(K) = m$, and hence the system matrix has full row rank md_v. Solvability thus requires $md_v \le d_k d_v$, or equivalently $m \le d_k$. This matches classic results on the storage capacity of linear associative memories such as the Willshaw model and Hopfield networks, where capacity is tied to the rank of the input embedding (Hopfield 1982; Willshaw et al. 1969).

When $m \le d_k$ and K has full column rank, one can construct an exact interpolating solution via the Moore–Penrose pseudoinverse: $\mathcal{M}^* = VK^\top$. Then $\mathcal{M}^*K = VK^\top K = V$, achieving zero training error. Thus the upper bound is tight.

Moreover, full-batch gradient descent on this objective with step size $0 < \eta < 2/\lambda_{\max}(KK^\top)$ yields iterates $\mathcal{M}_{t+1} = \mathcal{M}_t - \eta(\mathcal{M}_t K - V)K^\top$, which converge to the minimum-norm interpolating solution $\mathcal{M}^\dagger = VK^\top$ when $m \le d_k$. This is a well-known implicit bias of gradient descent in overparameterized linear models (Satpathi et al. 2021).

Finally, the same rank-based constraint governs the capacity of linear or multi-head attention modules. In such architectures, the output context matrix has rank at most $\text{rank}(K) \le d_k$, which directly limits their expressivity. Recent analyses identify this "low-rank bottleneck" as a capacity-limiting effect in Transformers (Bhojanapalli et al. 2020). □

Theorem 1 (Effect of Deep Memory)**.** *Let $\mathcal{M}(\cdot)$ be an MLP with $\mathcal{L}_{\mathcal{M}} \ge 2$ layers, d_k input dimension, and d_h hidden dimension. Then, $\mathcal{M}(\cdot)$ can store the mapping of at least $O(d_k d_v)$ and at most $O\left(d_k d_v \sum_{i=1}^{\mathcal{L}_{\mathcal{M}}} \min\{d_h^{(j)}\}_{j \ge i} d_h^{(j+1)}\right)$ pairs of $(\mathbf{k}_i, \mathbf{v}_i)$ with linearly independent keys.*

Early theoretical works established that even simple network architectures can memorize a significant number of input-output mappings, with capacity often related to the number of network parameters (e.g., weights and biases) and the input dimensionality Baum (1988), Cover (1965), and Huang (2003). For instance, Baum (1988) demonstrated that $\lceil \frac{N}{d} \rceil$ neurons are sufficient for a single-hidden-layer network with threshold units to memorize N input-label pairs from $\mathbb{R}^d$.

Networks employing Rectified Linear Units (ReLUs), exhibit a piecewise affine behavior. The input space is partitioned into numerous linear regions, and within each region, the network computes a distinct affine transformation Montufar et al. (2014) and Pascanu et al. (2014). This structure is pivotal for analyzing their expressive power and storage capacity. The precise relationship between depth, width, the number of linear regions, and the ultimate capacity to store specific key-value associations, especially with constraints like linearly independent keys, remains an active area of research.

Proof. Let m denote the number of $(\mathbf{k}_i, \mathbf{v}_i)$ pairs memorized exactly by $\mathcal{M}$, and assume the keys $\{\mathbf{k}_i\}_{i=1}^m \subset \mathbb{R}^{d_k}$ are linearly independent. Let $d_h^{(0)} := d_k$, $d_h^{(\mathcal{L}_\mathcal{M})} := d_v$, and for each layer $1 \le \ell \le \mathcal{L}_\mathcal{M}$, define $W^{(\ell)} \in \mathbb{R}^{d_h^{(\ell)} \times d_h^{(\ell-1)}}$. Biases are omitted for simplicity.

Since $\sigma(x) = \max(0, x)$ is piecewise linear, the composition of linear maps and ReLU activations yields a piecewise affine function. For any fixed activation pattern (i.e., fixed sign of pre-activations), the MLP acts as:

$$\mathcal{M}(\cdot) = A \cdot + B, \quad \text{where } A = W^{(\mathcal{L}_\mathcal{M})} D^{(\mathcal{L}_\mathcal{M}-1)} W^{(\mathcal{L}_\mathcal{M}-1)} \cdots D^{(1)} W^{(1)},$$

and each $D^{(\ell)}$ is a diagonal $\{0, 1\}$ matrix selecting the active units. Therefore, when all keys fall into the same linear region (which occurs generically after a small perturbation), $\mathcal{M}$ is a single affine transformation.

Let $\mathbf{K} := [\mathbf{k}_1 \ \cdots \ \mathbf{k}_m] \in \mathbb{R}^{d_k \times m}$ and $\mathbf{V} := [\mathbf{v}_1 \ \cdots \ \mathbf{v}_m] \in \mathbb{R}^{d_v \times m}$. Exact memorization implies $A\mathbf{K} = \mathbf{V}$, so:

$$\operatorname{rank}(\mathbf{V}) \le \operatorname{rank}(A), \quad m = \operatorname{rank}(\mathbf{K}) \le \min\{\operatorname{rank}(A), d_k\}.$$

Now observe:

$$A = W^{(\mathcal{L}_\mathcal{M})} \underbrace{D^{(\mathcal{L}_\mathcal{M}-1)} W^{(\mathcal{L}_\mathcal{M}-1)}}_{R_{\mathcal{L}_\mathcal{M}-1}} \cdots \underbrace{D^{(1)} W^{(1)}}_{R_1},$$

and thus the rank of A is bounded by the minimal width encountered along each path times the immediate input dimension:

$$\operatorname{rank}(A) \le \sum_{i=1}^{\mathcal{L}_\mathcal{M}} \left(\min_{j \ge i} d_h^{(j)} \right) d_h^{(i)} = O\left(d_k d_v \sum_{i=1}^{\mathcal{L}_\mathcal{M}} \min_{j \ge i} d_h^{(j)} d_h^{(i+1)} \right).$$

Hence,

$$m \le O\left(d_k d_v \sum_{i=1}^{\mathcal{L}_\mathcal{M}} \min_{j \ge i} d_h^{(j)} d_h^{(i+1)} \right)$$

□

Proposition 2 (Memory Capacity with Polynomial Mapping). *Let $\phi_p(\cdot)$ be a polynomial mapping with degree at most p, and $\mathcal{M}$ be a matrix-valued memory that optimizes the internal objective of $\ell(\mathcal{M}_t; \phi_p(\mathbf{k}_t), \mathbf{v}_t) = \|\mathcal{M}_t \phi_p(\mathbf{k}_t) - \mathbf{v}_t\|_2^2$ with gradient descent. $\mathcal{M}$ can store the mapping of at most $O\left(d_k{}^p\right)$ pairs of $(\mathbf{k}_i, \mathbf{v}_i)$ with linearly independent keys, where d_k is the dimension of keys $\mathbf{k}_i$.*

Proof. Let us begin by analyzing the dimension of the lifted feature space induced by ϕ_p. A monomial in d_k variables of total degree exactly ℓ has the form $\mathbf{k}^\alpha = \prod_{j=1}^{d_k} k_j^{\alpha_j}$, where $\alpha \in \mathbb{N}^{d_k}$ and $|\alpha| := \sum_{j=1}^{d_k} \alpha_j = \ell$. The number of such monomials is given by the classical stars-and-bars formula, which counts the number of integer solutions to $\alpha_1 + \cdots + \alpha_{d_k} = \ell$, yielding

$$\binom{d_k + \ell - 1}{\ell}.$$

Summing over all degrees $\ell = 0$ to p gives the total number of monomials (i.e., the output dimension of ϕ_p),

$$D = \sum_{\ell=0}^{p} \binom{d_k + \ell - 1}{\ell} = \binom{d_k + p}{p},$$

where the final identity follows from the hockey-stick identity in combinatorics.

To characterize the memorization capacity, we reformulate the loss in matrix notation. Let $\Phi := [\phi_p(\mathbf{k}_1) \ \cdots \ \phi_p(\mathbf{k}_m)] \in \mathbb{R}^{D\times m}$ and $V := [\mathbf{v}_1 \ \cdots \ \mathbf{v}_m] \in \mathbb{R}^{d_v \times m}$. Then the objective becomes

$$L(\mathcal{M}) = \tfrac{1}{2}\|\mathcal{M}\Phi - V\|_2^2.$$

Exact memorization corresponds to the existence of a matrix $\mathcal{M}$ such that $\mathcal{M}\Phi = V$. This is a linear system in which $\mathcal{M}$ acts on the columns of Φ, so the rank of Φ necessarily limits the number of independent targets $\mathbf{v}_i$ that can be fitted exactly.

By the sub-multiplicativity of rank, for any matrices A and B, we have

$$\operatorname{rank}(AB) \le \min\{\operatorname{rank}(A), \operatorname{rank}(B)\}.$$

Applying this to $\mathcal{M}\Phi$ yields

$$\operatorname{rank}(\mathcal{M}\Phi) \le \operatorname{rank}(\Phi) \le D.$$

Now consider a case where the targets $\mathbf{v}_1, \ldots, \mathbf{v}_m$ are linearly independent; for instance, take $V = [e_1, \ldots, e_m]$, the first m standard basis vectors. Then $\operatorname{rank}(V) = m$. If $m > D$, we necessarily have $\operatorname{rank}(\mathcal{M}\Phi) < \operatorname{rank}(V)$ for every choice of $\mathcal{M}$, implying that the system $\mathcal{M}\Phi = V$ is unsolvable. Hence, the loss remains strictly positive, and exact memorization is impossible.

This establishes that no method, regardless of optimization procedure, can memorize more than $D = \binom{d_k+p}{p}$ independent input-output pairs under a degree-$\le p$ polynomial lifting. Since $\binom{d_k+p}{p} = \Theta(d_k^p)$ for fixed p, the result follows: the memorization capacity is bounded above by $O(d_k^p)$. $\square$

D Detailed Formulations of All Architectures

In this section, for the sake of clarity, we discuss the details of all architectures that we discuss through the paper:

D.1 Deep Linear Attention (DLA)

We design Deep Linear Attention (DLA)—linear attention module that uses a deep MLP as the memory (KV cache)—as one of the baselines of this study. Given input $\mathbf{x} \in \mathbb{R}^{N\times d_{\text{in}}}$, we project the input into matrices of keys, values and queries:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{q}_1 \\ \vdots \\ \mathbf{q}_N \end{pmatrix} = \mathbf{x}\mathbf{W}_Q, \qquad \mathbf{K} = \begin{pmatrix} \mathbf{k}_1 \\ \vdots \\ \mathbf{k}_N \end{pmatrix} = \mathbf{x}\mathbf{W}_K, \qquad \mathbf{V} = \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \end{pmatrix} = \mathbf{x}\mathbf{W}_V, \tag{48}$$

where $\mathbf{W}_Q, \mathbf{W}_K$, and $\mathbf{W}_V$ are learnable linear layers. We then define memory as a learning module that optimizes the inner-dot product similarity using gradient descent: i.e.,

$$\min_{\mathcal{M}} \underbrace{\langle \mathcal{M}(\mathbf{k}_t), \mathbf{v}_t \rangle}_{\ell(\mathcal{M}_{t-1};\mathbf{k}_t,\mathbf{v}_t)}. \tag{49}$$

The above optimization using gradient descent results in the following recurrence (we also add weight decay with input-dependent parameter α_t):

$$\mathcal{M}_t = \alpha_t \mathcal{M}_{t-1} - \eta_t \nabla \ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t) \tag{50}$$

which in the case of linear memory (i.e., $\mathcal{M}_t = W_t \in \mathbb{R}^{d\times d}$) it becomes:

$$W_t = \alpha_t W_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top, \tag{51}$$

which is the formulation of gated linear attention. We use the same training process as other models (see Section 3.3).

D.2 Sliding Window Linear Attention (SWLA)

The design of SWLA is the same as the design of DLA, but with the use of sliding window objective. That is, given keys, values, and queries:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{q}_1 \\ \vdots \\ \mathbf{q}_N \end{pmatrix} = \mathbf{x}\mathbf{W}_Q, \qquad \mathbf{K} = \begin{pmatrix} \mathbf{k}_1 \\ \vdots \\ \mathbf{k}_N \end{pmatrix} = \mathbf{x}\mathbf{W}_K, \qquad \mathbf{V} = \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \end{pmatrix} = \mathbf{x}\mathbf{W}_V, \tag{52}$$

we optimize the internal objective of:

$$\min_{\mathcal{M}} \underbrace{\sum_{i=t-c+1}^{t} \langle \mathcal{M}_{t-1}(\mathbf{k}_i), \mathbf{v}_i \rangle}_{\ell(\mathcal{M}_{t-1};\mathbf{k}_t,\mathbf{v}_t)}. \tag{53}$$

The above formulation, results in:

$$\mathcal{M}_t = \alpha_t \mathcal{M}_{t-1} - \nabla \ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t) = \alpha_t \mathcal{M}_{t-1} - \sum_{i=t-c+1}^{t} \eta_i^{(t)} \nabla \langle \mathcal{M}_{t-1}(\mathbf{k}_i), \mathbf{v}_i \rangle, \tag{54}$$

which in the case of linear memory (i.e., $\mathcal{M}_t = W_t \in \mathbb{R}^{d \times d}$) it becomes:

$$\mathcal{M}_t = \alpha_t \mathcal{M}_{t-1} - \sum_{i=t-c+1}^{t} \eta_i^{(t)} \mathbf{v}_i \mathbf{k}_i^{\top}. \tag{55}$$

D.3 OmegaNet

In the design of OmegaNet, we use replace the dot-prodcut similarity objective with $\ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t) = \sum_{i=t-c+1}^{t} \|\mathcal{M}_{t-1}(\phi(\mathbf{k}_i)) - \mathbf{v}_i\|_2^2$,which results in the recurrence of:

$$\mathcal{M}_t = \alpha_t \mathcal{M}_{t-1} - \nabla \ell(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t) = \alpha_t \mathcal{M}_{t-1} - \sum_{i=t-c+1}^{t} \eta_i^{(t)} \nabla \|\mathcal{M}_{t-1}(\phi(\mathbf{k}_i)) - \mathbf{v}_i\|_2^2. \tag{56}$$

In the above formulation, $\phi(.)$ is the polynomial feature mapping function.

D.4 Atlas

In the Atlas, we use the same internal objective as OmegaNet but we optimize it using Muon optimizer (Jordan et al. 2024) with weight decay. That is,

$$\mathcal{M}_t = \alpha_t \mathcal{M}_{t-1} + \texttt{Newton-schulz5}(\mathcal{S}_t) \tag{57}$$

$$\mathcal{S}_t = \theta_t \mathcal{S}_{t-1} - \sum_{i=t-c+1}^{t} \eta_i^{(t)} \nabla \|\mathcal{M}_{t-1}(\phi(\mathbf{k}_i)) - \mathbf{v}_i\|_2^2. \tag{58}$$

E Experimental Details

In our experimental setup we follow recent studies on linear recurrent models (Behrouz, Razaviyayn, et al. 2025; Behrouz, Zhong, et al. 2024; Yang, Kautz, et al. 2024), we use Wikitext (Merity et al. 2017), LMB (Paperno et al. 2016), PIQA (Bisk et al. 2020), HellaSwag (Zellers et al. 2019), WinoGrande (Sakaguchi et al. 2021), ARC-easy (ARC-e) and ARC-challenge (ARC-c) (Clark, Cowhey, et al. 2018), SIQA (Sap et al. 2019), and BoolQ (Clark, Lee, et al. 2019). Also, the baselines results are from Behrouz, Razaviyayn, et al. (2025) and Behrouz, Zhong, et al. (2024). In the training, we use T5 tokenizer with a vocabulary size of 32K and use training length of 4K tokens (2K for SWA). We employ AdamW optimizer with learning rate of 4*e*-4 with cosine annealing schedule with batch size of 0.5M tokens, and weight decay of 0.1. The architectural

Table 7: Architectural Details.

Model	Block	Dim	Head	Peak LR	Token
170M	12	768	16	3e-3	15B
340M	24	1024	16	1.5e-3	15B
760M	24	1536	16	1.25e-3	30B
1.3B	18	2048	8	7e-4	100B

details are also reported in Table 7. The baseline results for 1.3B are from Yang, Kautz, et al. (2024) and for 760M are from Behrouz, Razaviyayn, et al. (2025) and Behrouz, Zhong, et al. (2024).

For the memory architecture, unless state otherwise, we use an MLP with 2 layers with expansion factor of 4 and GELU activation function (Hendrycks et al. 2016). We also use residual connections and layer norm at the end of each chunk: $\mathcal{M}(x) = x + W_1\sigma(W_2 x)$.

F Additional Experimental Results

In this section, we provide additional experimental results to support the design of our models, understand the effect of different components and also evaluate their performance in long context, in-context recall and MAD tasks.

F.1 Language Modeling and Common-sense Reasoning (Small Scale)

In Section 6 we presented a subset of results on language modeling and common-sense reasoning tasks. In this section, we further report the results for all scales of models. The results are in Table 8.

State-of-the-art Results. Looking at the performance of Atlas and OmegaNet, both architectures perform favorably compared to modern linear recurrent models and Transformers, achieving lower perplexity and better accuracy in downstream tasks. Even the fully recurrent version of these models outperform hybrid models such as Samba (Ren et al. 2024) and Gated DeltaNet-H2 (Yang, Kautz, et al. 2024). Using the hybrid variants of MAG and MAL further improve the performance of Atlas, which shows the complementary role of recurrent long-term memory and attention.

The Effect of Design. Comparing the performance of Atlas, OmegaNet, and baselines SWLA and DLA, we can see the role of ℓ_2 regression loss as the attentional bias. Also, the better performance of SWLA compared to GLA and RetNet indicates the importance of memorizing the context, instead of memorizing individual tokens.

Table 8: Performance of ATLAS and baselines on language modeling and common-sense reasoning tasks. The best results are highlighted highlighted .

Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg. ↑
340M params / 15B tokens											
Transformer++	31.52	41.08	30.76	62.98	34.76	50.53	45.21	24.05	36.81	58.24	42.92
RetNet	32.50	49.73	28.24	62.61	34.15	50.91	44.27	23.62	36.79	59.72	42.54
GLA	28.51	43.02	28.73	64.05	35.96	50.00	54.19	24.29	37.13	58.39	44.09
Mamba	30.83	40.21	29.94	63.79	35.88	49.82	49.24	24.56	35.41	60.07	43.59
DeltaNet	28.65	47.30	28.43	63.52	35.95	49.63	52.68	25.37	37.96	58.79	44.04
TTT	27.44	34.19	30.06	63.97	35.71	50.08	53.01	26.11	37.32	59.83	44.51
Gated DeltaNet	27.01	30.94	34.11	63.08	38.12	51.60	55.28	26.77	34.89	59.54	45.42
MONETA	26.19	29.31	35.70	63.99	39.23	52.04	55.96	27.15	37.29	60.22	46.44
YAAD	26.61	29.11	34.09	64.93	39.86	51.12	54.75	28.64	33.82	60.29	45.93
MEMORA	27.16	30.44	33.68	65.21	39.17	51.23	53.40	27.99	34.1	59.29	45.51
DLA (ours)	27.93	35.09	30.8	62.9	36.2	50.4	53.5	26.7	37.1	59.7	44.76
SWDT (ours)	26.98	33.95	32.4	63.1	38.2	50.9	54.9	25.9	37.5	59.6	45.31
OMEGANET (ours)	26.03	28.76	35.6	65.3	39.7	52.0	56.1	28.6	37.7	60.4	46.93
ATLAS (ours)	25.88	28.54	36.1	64.9	40.1	52.7	56.4	28.8	38.1	61.2	47.28