Nested Learning: The Illusion of Deep Learning Architectures

Ali Behrouz
Google Research
USA
alibehrouz@google.com

Meisam Razaviyayn
Google Research
USA
razaviyayn@google.com

Peiling Zhong
Google Research
USA
peilinz@google.com

Vahab Mirrokni
Google Research
USA
mirrokni@google.com

Abstract

Over the last decades, developing more powerful neural architectures and simultaneously designing optimization algorithms to effectively train them have been the core of research efforts to enhance the capability of machine learning models. Despite the recent progresses, particularly in developing Language Models (LMs), there are fundamental challenges and unanswered questions about how such models can *continually learn/memorize, self-improved, and find "effective solutions,"*. In this paper, we present a new learning paradigm, called Nested Learning (NL), that coherently represents a model with a set of nested, multi-level, and/or parallel optimization problems, each of which with its own "*context flow*". NL reveals that existing deep learning methods learns from data through *compressing* their own context flow, and explain how *in-context learning* emerges in large models. NL suggests a path (a new dimension to deep learning) to design more expressive learning algorithms with more "*levels*", resulting in higher-order in-context learning abilities. In addition to its neuroscientifically plausible and mathematically white-box nature, we advocate for its importance by presenting three core contributions: (1) Deep Optimizers: Based on NL, we show that well-known gradient-based optimizers (e.g., Adam, SGD with Momentum, etc.) are in fact associative memory modules that aim to compress the gradients with gradient descent. Building on this insight, we present a set of more expressive optimizers with deep memory and/or more powerful learning rules; (2) Self-Modifying Titans: Taking advantage of NL's insights on learning algorithms, we present a novel sequence model that learns how to modify itself by learning its own update algorithm; and (3) Continuum Memory System: We present a new formulation for memory system that generalizes the traditional viewpoint of "long-term/short-term memory". Combining our self-modifying sequence model with the continuum memory system, we present a learning module, called HOPE, showing promising results in language modeling, continual learning, and long-context reasoning tasks.

1 Introduction

This version of the paper has been extensively summarized to fit the page limit of NeurIPS camera ready, and some materials, experiments, discussions, and methods are moved to appendix, which might make some parts hard to follow or cause inconsistencies. To avoid such cases, please read our arXiv version instead [1] (will be available on November 13).

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

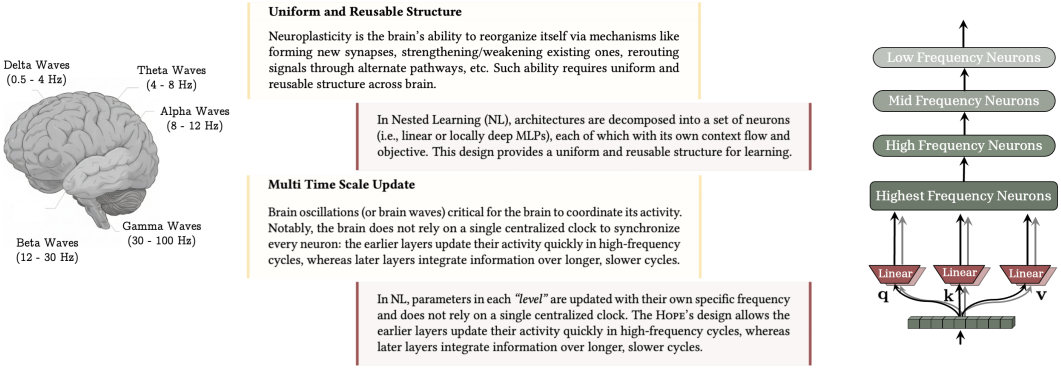

Figure 1: The uniform and reusable structure as well as multi time scale update in the brain are the key components to unlock the continual learning in humans. Nested Learning (NL) allows for multi time-scale update for each component of the brain, while showing that well-known architectures such as Transformers are in fact linear layers with different frequency updates.

For decades, AI research has focused on designing machine learning algorithms that learn from data [2–5] or experience [6–8]; often by optimizing an objective $\mathcal{L}(\boldsymbol{\theta})$ over parameters $\boldsymbol{\theta} \in \Theta$ with gradient-based methods. While traditional machine learning techniques required careful engineering and domain expertise to design feature extractors, limiting their ability to directly process and learn from natural data [9], deep representation learning offered a fully automated alternative to discover the representations needed for the task. Thereafter, deep learning has been an inseparable part of the large-scale computational models with seminal success in chemistry and biology [10], games [11, 12], computer vision [13, 14], and multimodal and natural language understanding [15–17].

Stacking of multiple layers, as it is done in deep learning models, provides the models with larger capacity, better expressive power in representing complex features, and more internal computations (e.g., #FLOPS) [18–20], all of which are critical and desirable characteristics for static tasks that require in-distribution predictions over a previously fixed set. This deep design, however, is not a universal solution to all the challenges and cannot help the expressive power of the models in multiple aspects, for example: (i) The computational depth of deep models might not change with more layers [21, 22], leaving their ability to implement complex algorithms untouched compared to traditional shallow approaches [23]; (ii) The capacity of some class of parameters might show marginal improvement with increasing the depth/width of the model [24]; (iii) The training process might converge to a suboptimal solution, mainly due to the suboptimal choice of the optimizer or its hyperparameters; and (iv) The model's ability to fast adapt to a new task, continually learn, and/or generalize to out-of-distribution data might not changed with stacking more layers and requires more careful designs.

The core part of the efforts to overcome the above challenges and to enhance the capability of deep learning models concentrate on: (1) developing more expressive class of parameters (i.e., neural architectures) [13, 25–28]; (2) introducing objectives that can better model the tasks [29–32]; (3) designing more efficient/effective optimization algorithms to find better solutions or with more resilience to forgetting [33–36]; and (4) scaling the model size to enhance its expressivity, when the "right" choice of architecture, objective, and optimization algorithms are made [24, 37, 38]. Collectively, these advancements and new findings on scaling patterns of deep models have established the foundations upon which Large Language Models (LLMs) have been built.

The development of LLMs marks a pivotal milestone in deep learning research: a paradigm shift from task-specific models to more general-purpose systems with various emergent capabilities as a result of scaling the "right" architectures [38, 39]. Despite all their success and remarkable capabilities in diverse sets of tasks [15, 40, 41], LLMs are largely static after their initial deployment phase, meaning that they successfully perform tasks learned during pre- or post-training, but are unable to continually acquire new capabilities beyond their immediate context. The only adaptable component of LLMs is their *in-context learning* ability–a (known to be emergent) characteristic of LLMs that enables fast adaption to the context and so perform zero- or few-shot tasks [38]. Beyond in-context learning, recent efforts to overcome the static nature of LLMs either are computationally expensive, require external components, lack generalization, and/or might suffer from catastrophic forgetting [42–44], which has led researchers to question if there is a need to revisit how to design machine learning

models and if a new learning paradigm beyond stacking of layers is required to unleash the capabilities of LLMs in continual setups.

Current Models only Experience the Immediate Present. As an analogy and to better illustrate the static nature of LLMs, we use the example of anterograde amnesia–a neurological condition where a person cannot form new long-term memories after the onset of the disorder, while existing memories remain intact [45]. This condition limits the person's knowledge and experiences to a short window of present and long past–before the onset of the disorder–which results in continuously experiencing the immediate present as if it were always new. The memory processing system of current LLMs suffer from a similar pattern. Their knowledge is limited to either, the immediate context that fits into their context window, or the knowledge in MLP layers that stores long-past, before the onset of *"end of pre-training."* This analogy, has motivated us to take inspiration from neurophysiology literature and how brain consolidate its short-term memories:

1.1 Human Brain Perspective and Neurophysiological Motivation

Human brain is highly efficient and effective when it comes to continual learning (a.k.a. effective context management), which is often attributed to neuroplasticity—the brain's remarkable capacity to change itself in response to new experiences, memories, learning, and even damage [46, 47]. Recent studies support that the formation of Long-term memory involves at least two distinct but complementary consolidation processes [48–50]: (1) A rapid "online" consolidation (also known as synaptic consolidation) phase occurs immediately or soon after learning, even during wakefulness. This is when new and initially fragile memory traces are stabilized and begin transferring from short-term to long-term storage; (2) An "offline" consolidation (also known as systems consolidation) process repeats the replay of the recently encoded patterns—during sharp-wave ripples (SWRs) in the hippocampus, coordinated with cortical sleep spindles and slow oscillations—strengthens and reorganizes the memory and supports transfer to cortical sites [51–53].

Coming back to the analogy of anterograde amnesia, evidence indicates that the condition can impact both stages, but especially the online consolidation phase, mainly due to the fact that hippocampus is the gateway for encoding new declarative memories, and so its damage means new information never will be stored in long-term memory. As mentioned above, the design of LLMs, and more specifically Transformer-based backbones, suffers from a similar condition after the pre-training phase. That is, the information provided in the context, never impacts the long-term memory parameters (e.g., feedforward layers), and so the model is not capable of acquiring new knowledge or skill, unless the information is still stored in the short-term memory (e.g., attention). To this end, although the second stage is equally, or even more, crucial for the consolidation of memories, and its absence can damage the process and might cause loss of memory [54, 55], in this work, we focus on the first stage: memory consolidation as an online process. We provide additional discussion on human brain perspective and its connection to NL in Appendix A.

Notations. We let $x \in \mathbb{R}^{N \times d_{\text{in}}}$ be the input, $\mathcal{M}_t$ represent the state of memory/model $\mathcal{M}$ 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 the vector corresponds to the input t (i.e., $\mathbf{k}_t, \mathbf{v}_t$, and $\mathbf{q}_t$). We further refer to the distribution of any entities f as $p(f)$. 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 $\mathcal{M}(\cdot)$. When it is needed, we parameterized the memory module with $\boldsymbol{\theta}_{\mathcal{M}} \supseteq \{W_1, W_2, \ldots, W_{\mathcal{L}_{\mathcal{M}}}\}$, which at least includes the parameters of linear layers in the MLP. We use superscript with parenthesis to refer to parameters in different *levels* of nested learning (different update frequency): i.e., $W^{(\ell)}$.

2 Nested Learning

This section discusses the motivations, formal definitions, and general high-level implications of Nested Learning (NL). We start with a formulation of associative memory and then by using step-by-step examples, we build the intuition behind architecture decomposition and its connection to modeling a neural network as an integrated system of optimization problems. We aim to first show how existing methods and concepts in deep learning fall under the NL paradigm and then we present new formulations that go beyond traditional methods and/or provide insights on how to improve existing algorithms and designs.

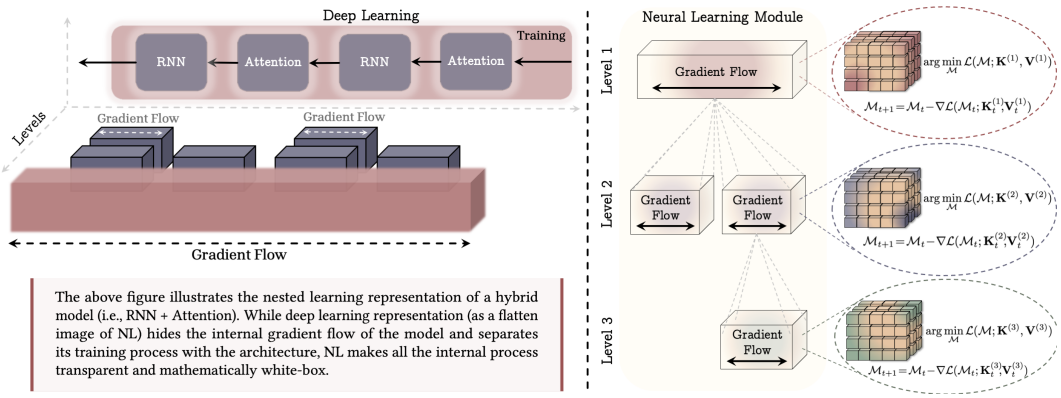

Figure 2: Nested Learning Paradigm that represent a machine learning model and its training procedure as a set of nested optimization problems. (**Left**) An example of Hybrid architecture. While deep learning perspective, as the flattened image of NL, does not provide insight about the depth of computation in the blocks, NL transparently represent all the inner gradient flows. (**Right**) A Neural Learning Module: A computational model that learns how to compress its own context flow. For example, the first level corresponds to the model's the most outer-loop training, often refer to as "*pre-training*" step.

2.1 Associative Memory

Associative memory—the ability to form and retrieve connections between events—is a fundamental mental process and is an inseparable component of human learning [56]. Often in the literature, the concept of memorization and learning are used interchangeably; in neuropsychology literature, however, these two are clearly distinguished. More specifically, following neuropsychology literature [57], we build our terminology based on the following definition of memory and learning:

> **Learning vs. Memorization:**
>
> *Memory is a neural update caused by an input, and learning is the process for acquiring effective and useful memory.*

In this work, our goal is to first show that all the elements of a computational sequence model, including optimizers and neural networks, are *associative memory systems* that compress their own *context flow*. Broadly speaking, associative memory is an operator that maps a set of keys to a set of values. We follow the general definition of associative memory by Behrouz et al. [58]:

Definition 1 (Associative Memory)**.** *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 operator $\mathcal{M} : \mathcal{K} \to \mathcal{V}$ that maps two sets of keys $\mathcal{K}$ and values $\mathcal{V}$. To learn such mapping from the data, an objective $\tilde{\mathcal{L}}(\cdot;\cdot)$ measures the quality of the mapping and $\mathcal{M}$ can be defined as:*

$$\mathcal{M}^* = \arg\min_{\mathcal{M}} \quad \tilde{\mathcal{L}}(\mathcal{M}(\mathcal{K}); \mathcal{V}). \tag{1}$$

While the operator itself is a memory and the mapping acts as a memorization process (i.e., memorizing the connections of events in the context), acquiring such effective operator based on the data, is a learning process. It is notable that, here, keys and values can be any arbitrary events that memory aims to map them and are not limited to tokens. Later in this section, we will discuss that given a context flow, keys and values might be tokens, gradients, sub-sequences, etc. Furthermore, while the term of associative memory is more common in neuroscience and neuropsychology literature, the above formulation is also closely related to data compression and low-dimensional representation. That is, one can interpret the optimization process in Equation 1 as the training process of a network $\mathcal{M}(.)$ that aims to compress the mappings into its parameters and so represent them in a lower dimensional space.

In sequence modeling, where keys and values are input tokens (e.g., tokenized text), the choice of objective and the optimization process for solving Equation 1 can result in distinct sequence

modeling architectures (see [59] and [58]) such as global/local softmax attention [27], or other modern recurrent models [28, 60, 61]. This simple formulation of sequence models provides us with better understanding of their internal process and also a tool to simply compare their modeling power based on their objective and optimization process. In the following, using step-by-step examples, we discuss how this formulation can be applied to all components of a neural architecture (including its optimization process in pre-training) and in fact, how a model is an integrated system of multi-level, nested, and or parallel memories, each of which with its own context flow.

A Simple Example of MLP Training. We start with a simple example, in which we aim to train a 1-layer MLP (parameterized with W) for task $\mathcal{T}$ and on dataset $\mathcal{D}_{\text{train}} = \{x_1, \ldots, x_{|\mathcal{D}_{\text{train}}|}\}$ by optimizing the objective $\mathcal{L}(\cdot;\cdot)$ with gradient descent. In this case, the training process is equivalent to the following optimization problem:

$$W^* = \arg\min_{W} \; \mathcal{L}(W; \mathcal{D}_{\text{train}}), \tag{2}$$

whose optimization by gradient descent results in a weight update rule equivalent to:

$$\begin{aligned} W_{t+1} &= W_t - \eta_{t+1} \nabla_{W_t} \mathcal{L}(W_t; x_{t+1}) && (3)\\ &= W_t - \eta_{t+1} \nabla_{y_{t+1}} \mathcal{L}(W_t; x_{t+1}) \otimes x_{t+1}, \quad \text{where } x_{t+1} \sim \mathcal{D}_{\text{train}}, && (4) \end{aligned}$$

where $y_{t+1} = W x_{t+1}$ is the output of the model for input x_{t+1}. Given this formulation, one can let $u_{t+1} = \nabla_{y_{t+1}} \mathcal{L}(W_t; x_{t+1})$ and reformulate the backpropagation process as the solution to an optimization problem on finding an optimal associative memory that maps input data points $\mathcal{D}_{\text{train}} = \{x_t\}_{t=1}^{|\mathcal{D}_{\text{train}}|}$ to their corresponding $u_{t+1} = \nabla_{y_{t+1}} \mathcal{L}(W_t; x_{t+1})$. That is, we let $\mathcal{M}(\cdot) = W_t \cdot$ parametrizes the memory, and use dot-product similarity to measure the quality of W_t's mapping between x_{t+1} and $\nabla_{y_{t+1}} \mathcal{L}(W_t; x_{t+1})$:

$$\begin{aligned} W_{t+1} &= \arg\min_{W} \; \langle W x_{t+1}, u_{t+1} \rangle + \frac{1}{2\eta_{t+1}} \|W - W_t\|_2^2 && (5)\\ &= \arg\min_{W} \; \langle W x_t, \nabla_{y_{t+1}} \mathcal{L}(W_t; x_{t+1}) \rangle + \frac{1}{2\eta_{t+1}} \|W - W_t\|_2^2. && (6) \end{aligned}$$

In the above formulation, $u_{t+1} = \nabla_{y_{t+1}} \mathcal{L}(W_t; x_{t+1})$ can be interpreted as a *local surprise signal in representation space* that quantifies the mismatch between the current output and the structure the objective $\mathcal{L}(\cdot;\cdot)$ enforces. Therefore, this formulation translates the training phase of the model as a process of acquiring effective memory that maps data samples to their Local Surprise Signal (LSS) in representation space–defined as the mismatch between the current output and the structure enforced by the objective $\mathcal{L}(\cdot;\cdot)$. Accordingly, in this example, our model has *a single gradient flow* over the data samples, which is only active over dataset $\mathcal{D}_{\text{train}} = \{x_1, \ldots, x_{|\mathcal{D}_{\text{train}}|}\}$ and will be frozen for any other data samples afterwards (a.k.a inference or test time).

Next, in the above example, we replace the gradient descent algorithm with its enhanced momentum-based variant, resulting in an update rule of:

$$\begin{aligned} W_{t+1} &= W_t - \mathbf{m}_{t+1}, && (7)\\ \mathbf{m}_{t+1} &= \mathbf{m}_t - \eta_{t+1} \nabla_{W_t} \mathcal{L}(W_t; x_{t+1}) = \mathbf{m}_t - \eta_{t+1} \nabla_{y_{t+1}} \mathcal{L}(W_t; x_{t+1}) \otimes x_{t+1}. && (8) \end{aligned}$$

In Equation 8, given the previous state of Equation 7 (at time t), the value of $\nabla_{W_t} \mathcal{L}(W_t; x_{t+1})$ or similarly $\nabla_{y_{t+1}} \mathcal{L}(W_t; x_{t+1})$ are independent of recurrence in Equation 8 and so can be pre-computed beforehand. To this end, we let $u_{t+1} = \nabla_{W_t} \mathcal{L}(W_t; x_{t+1})$, and so Equation 8 can be reformulated as:

$$\begin{aligned} W_{t+1} &= W_t - \mathbf{m}_{t+1}, && (9)\\ \mathbf{m}_{t+1} &= \arg\min_{\mathbf{m}} \; -\langle \mathbf{m}, \nabla_{W_t} \mathcal{L}(W_t; x_{t+1}) \rangle + \eta_{t+1} \|\mathbf{m} - \mathbf{m}_t\|_2^2 && (10)\\ &= \arg\min_{\mathbf{m}} \; -\langle \mathbf{m}\, x_{t+1}, \nabla_{y_{t+1}} \mathcal{L}(W_t; x_{t+1}) \rangle + \eta_{t+1} \|\mathbf{m} - \mathbf{m}_t\|_2^2, && (11) \end{aligned}$$

where the optimization problem in Equation 10 is equivalent to on step of gradient descent with adaptive learning rate of η_{t+1}. Given these formulation, one can interpret the momentum term as either: (1) a key-less associative memory that compress the gradients into its parameters, or (2) an associative memory that learns how to map data points to their corresponding LSS-value. Interestingly, this formulation reveals that gradient descent with momentum is indeed a two-level

optimization process, where the memory is optimized by simple gradient descent algorithm. This process is closely related to Fast Weight Programs (FWPs) [62], where the weight update process (i.e., Equation 9) is the slow network that its momentum weight is generated by a fast network (i.e., Equation 10).

Concluding the above examples, we observed that the training process of a 1-layer MLP with: (1) Gradient descent is a *1-level* associative memory that learns how to map data points to their corresponding LSS-value; and (2) Gradient descent with momentum is a *2-level* associative memory (or optimization process) that the inner-level learns to store gradient values into its parameters, and then the outer-level updates the slow weight (i.e., W_t) with the value of the inner-level memory. While these are the most simple examples with respect to both architecture and optimizer algorithms, one might ask if similar conclusion can be made in more complex setups.

An Example of Architectural Decomposition. In the next example, we replace the MLP module with a linear attention [60]. That is, we aim to train a 1-layer linear attention for task $\mathcal{T}$ and on a sequence of $\mathcal{D}_{\text{train}} = \{x_1, \ldots, x_{|\mathcal{D}_{\text{train}}|}\}$ by optimizing the objective $\mathcal{L}$ with gradient descent. Recalling the unnormalized linear attention formulation:

$$\mathbf{k}_t = x_t W_{\mathbf{k}}, \qquad \mathbf{v}_t = x_t W_{\mathbf{v}}, \qquad \mathbf{q}_t = x_t W_{\mathbf{q}}, \tag{12}$$

$$\mathcal{M}_t = \mathcal{M}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top, \tag{13}$$

$$y_t = \mathcal{M}_t \mathbf{q}_t \, . \tag{14}$$

As discussed in earlier studies [58, 59], the recurrence in Equation 13 can be reformulated as the optimization process of a matrix-valued associative memory $\mathcal{M}_t(\cdot)$, in which, it aims to compress the mappings of keys and values into its parameters. In more details, in Definition 1, if we let $\tilde{\mathcal{L}}(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t) := -\langle \mathcal{M}_{t-1}\mathbf{k}_t, \mathbf{v}_t \rangle$ and aim to optimize the memory with gradient descent, the memory update rule is: (Note that $\nabla \tilde{\mathcal{L}}(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t) = \mathbf{v}_t \mathbf{k}_t^\top$ and we let learning rate $\eta_t = 1$)

$$\mathcal{M}_{t+1} = \arg\min_{\mathcal{M}} \; \langle \mathcal{M}\mathbf{k}_{t+1}, \mathbf{v}_{t+1} \rangle + \|\mathcal{M} - \mathcal{M}_t\|_2^2 \quad \text{with gradient descent,} \tag{15}$$

$$\Rightarrow \; \mathcal{M}_{t+1} = \mathcal{M}_t - \nabla \tilde{\mathcal{L}}(\mathcal{M}_t; \mathbf{k}_{t+1}, \mathbf{v}_{t+1}) = \mathcal{M}_t + \mathbf{v}_{t+1}\mathbf{k}_{t+1}^\top, \tag{16}$$

which is equivalent to the update rule of an unnormalized linear attention in Equation 13. Also, note that as we observed in the first example, training a linear layer with gradient descent is a 1-layer optimization problem of an associative memory (see Equation 3) and so the general training/updating process of projection layers (i.e., $W_{\mathbf{k}}, W_{\mathbf{v}}$, and $W_{\mathbf{q}}$) is itself an optimization process of associative memory. Accordingly, this setup, i.e., training a linear attention with gradient descent, can be seen as a two-level optimization process, where the outer-loop (also known as training process) optimizes the projection layers with gradient descent, while the inner-loop optimizes the inner memory of $\mathcal{M}_t$ with gradient descent.

Note that, as discussed above, here, we have two associative memories, and so each of which has their own optimization process and gradient flow. That is, in the optimization of outer-level parameters of $W_{\mathbf{k}}, W_{\mathbf{v}}$, and $W_{\mathbf{q}}$ there is no gradient with respect to parameter $\mathcal{M}(\cdot)$ and so there is no backpropagation through it. Similarly, in the inner-level, there is no backpropagation through projection layers and they are considered frozen. Furthermore, it is notable that in this example, the above formulation is also closely connected to FWPs perspective of linear attentions [63], where projections are considered slow weights, and memory update in Equation 13 is the fast weight update rule.

Architectural Decomposition with More Levels. In both above examples, we discussed simple cases, where they can be translated into 2-level optimization processes, which also coincides with their FWPs interpretations. In practice, however, we need to use more powerful optimization algorithms to train the model, and/or use more powerful recurrent update rule for memory. As a simple example, assume we use gradient descent with momentum to train a linear attention model. In the above examples, we show that how the linear attention component can be decomposed into two nested optimization problem. Similarly, here the model can be represented as a 2-level optimization problem, where (1) the inner level optimizes the memory to compress the context using gradient descent (see Equation 15), and (2) the outer level optimizes the projection layers with gradient descent with momentum. Interestingly, from the first example, we know that "gradient descent with momentum" algorithm itself is indeed a 2-level optimization problem where the momentum term itself is an associative memory that compress the past gradients into its parameters.

2.2 Nested Optimization Problems

In the previous section, we provided examples to demonstrate how one can decompose a machine learning model into a set of nested or multi-level optimization problems. Next, we first aim to present a formal formulation for nested learning problems and then define Neural Learning Module–an integrated computational system that learns from data.

As we observed in the previous section, while we decomposed the model into a set of optimization process, it is still unclear if we can define a hierarchy (or order) over these problems, and uniquely represent the model in this format. Inspired by the hierarchy of brain waves that indicates the information processing frequency rate of each part (discussed in Section 1), we use the update rate of each optimization problem to order the components in multiple levels. To this end, we let the one update step over one data point to be the unit of time, and define the update frequency rate of each component as:

Definition 2 (Update Frequency)**.** *For any component of A, which can be a parametric component (e.g., learnable weights or momentum term in gradient descent in momentum) or a non-parametric component (e.g., attention block), we define its frequency, denoted as f_A, as its number of updates per unit of time.*

Given the above update frequency, we can order the components of a machine learning algorithm based on operator $(\cdot \succ \cdot)$. We let A to be faster than B and denote $A \succ B$ if: **(1)** $f_A > f_B$, or **(2)** $f_A = f_B$ but the computation of the B's state at time t requires the computation of A's state at time t. In this definition, when $A \not\succ B$ and $B \not\succ A$, we let $A \overset{f}{=} B$, which indicates that A and B has the same frequency update, but their computation is independent of each other (Later, we provide an example of this cases in AdamW optimizer). Based on the above operator, we sort the components into an ordered set of "*levels*", where (1) components in the same level have the same frequency update, and (2) the higher the level is, the lower its frequency. Notably, based on the above definition, each component has its own optimization problem and so context. While we optimize the component's inner objective with gradient-based optimizers, the above statement is equivalent to having exclusive gradient flow for each component in the model. In general case, however, one can use non-parametric solution (as we later discuss about attention).

Neural Learning Module. Given the above definition of nested learning problems, we define neural learning module as a new way of representation of machine learning models that shows the model as an interconnected system of components, each of which with its own gradient flow. Note that, orthogonal to deep learning, nested learning allows us to define neural learning models with more levels, resulting in more expressive architecture.

> ***Nested learning*** *allows computational models that are composed of multiple (multi-layer) levels to learn from and process data with different levels of abstraction and time-scales.*

Next, we study optimizers and well-known deep learning architectures from the nested learning perspective, and provide examples that how NL can help to enhance those components.

2.3 Optimizers as Learning Modules

In this section, we start by understanding how well-known optimizers and their variants are special instances of nested learning. Recall the gradient descent method with momentum,

$$\begin{aligned} W_{i+1} &= W_i + \mathbf{m}_{i+1} \\ \mathbf{m}_{i+1} &= \alpha_{i+1}\mathbf{m}_i - \eta_t \nabla \mathcal{L}\left(W_i; x_i\right), \end{aligned} \tag{17}$$

where matrix (or vector) $\mathbf{m}_i$ is the momentum at state i and α_i and η_i are adaptive learning and momentum rates, respectively. Assuming $\alpha_{i+1} = 1$, the momentum term can be viewed as the result of optimizing the following objective with gradient descent:

$$\min_{\mathbf{m}} \ \langle \mathbf{m}\, \nabla \mathcal{L}(W_i; x_i)^\top, \mathbf{I} \rangle. \tag{18}$$

This interpretation shows that momentum can indeed be viewed as a meta memory module that learns how to memorize gradients of the objective into its parameters. Building on this intuition, in

Section C.4 we show that Adam with a small modification is the optimal associative memory for the models' gradients. Next, we show that how this perspective can result in designing more expressive optimizers:

Extension: More Expressive Association. As discussed earlier, momentum is a value-less associative memory and so has limited expressive power. To address this issue, following the original definition of associative memory (i.e., mapping keys to values), we let value parameter $\mathbf{v}_i = \mathbf{P}_i$ and so the momentum aims to minimize:

$$\min_{\mathbf{m}} \ \langle \mathbf{m}\, \nabla \mathcal{L}(W_i; x_i)^\top, \mathbf{P}_i \rangle, \tag{19}$$

using gradient descent, resulting in the update rule:

$$\begin{aligned} W_{i+1} &= W_i + \mathbf{m}_{i+1} \\ \mathbf{m}_{i+1} &= \alpha_{i+1} \mathbf{m}_i - \eta_t \mathbf{P}_i \nabla \mathcal{L}\left(W_i; x_i\right). \end{aligned} \tag{20}$$

This formulation is equivalent to using preconditioning the momentum GD. In fact, preconditioning means that the momentum term is an associative memory that learns how to compress the mappings between $\mathbf{P}_i$ and the gradient term $\nabla \mathcal{L}(W_i; x_i)$. While any reasonable choice (e.g., random features) of preconditioning can improve the expressivity of the initial version of GD with momentum per se is a value-less memory (i.e., mapping all gradients to a single value), the above perspective gives more intuition about what preconditioning are more useful. That is, the momentum acts as a memory that aims to map gradients to their corresponding values, and so a function of gradients (e.g., information about Hessian) can provide the memory with a more meaningful mappings.

Extension: More Expressive Objectives. As discussed by Behrouz et al. [58], optimizing an inner objective of dot-product similarity results in Hebbian-like update rule, which can cause the memory to be less effective. A natural extension of this internal objective is to use $\ell_2(\cdot)$ regression loss (for measuring the corresponding key-value mapping fitness) and minimize the loss function $\|\mathbf{m} \nabla \mathcal{L}(W_i; x_i)^\top - \mathbf{P}_i\|_2^2$, resulting in the update rule of:

$$W_{i+1} = W_i + \mathbf{m}_{i+1}, \tag{21}$$

$$\mathbf{m}_{i+1} = \left(\alpha_{i+1} \mathbf{I} - \nabla \mathcal{L}\left(W_i; x_i\right)^\top \nabla \mathcal{L}\left(W_i; x_i\right)\right) \mathbf{m}_i - \eta_t \mathbf{P}_i \nabla \mathcal{L}\left(W_i; x_i\right), \tag{22}$$

This update is based on delta-rule [64] and so it allows the memory (momentum) to better manage its limited capacity and better memorize the series of past gradients.

Extension: More Expressive Memory. As discussed earlier, momentum can be viewed as a meta memory model that uses a linear layer (i.e., matrix-valued) to compress the past gradient values. Due to the linear nature of momentum, only linear functions of past gradients can be learned by its internal objective. To increase the learning capacity of this module, one alternative is to use alternative powerful persistent learning modules: i.e., replacing a linear matrix-valued memory for momentum with an MLP. Therefore, momentum as the a memory for the past gradients, has more capacity to capture the underlying dynamics of the gradients. To this end, we extend the formulation in Equation 17 as:

$$W_{i+1} = W_i + \mathbf{m}_{i+1}\left(\mathbf{u}_i\right), \quad \text{and} \quad \mathbf{m}_{i+1} = \alpha_{i+1} \mathbf{m}_i - \eta_t \nabla \mathcal{L}^{(2)}(\mathbf{m}_i; \mathbf{u}_i, \mathbf{I}), \tag{23}$$

where $\mathbf{u}_i = \nabla \mathcal{L}\left(W_i; x_i\right)$ and $\nabla \mathcal{L}^{(2)}(\cdot)$ is the internal objective of momentum (e.g., dot product similarity $\langle \mathbf{m}(\mathbf{u}_i^\top), \mathbf{1} \rangle$). We refer to this variant as Deep Momentum Gradient Descent (DMGD).

Extension: None Linear Outputs. Building upon the above perspective, in which we see the momentum as a neural architecture, one common technique to enhance the representation power of momentum memory module is to use non-linearity on top of its output [28, 65]. That is, we re-formulate Equation 23 as:

$$W_{i+1} = W_i + \sigma\left(\mathbf{m}_{i+1}\left(\mathbf{u}_i\right)\right), \quad \text{and} \quad \mathbf{m}_{i+1} = \alpha_{i+1} \mathbf{m}_i - \eta_t \nabla \mathcal{L}^{(2)}(\mathbf{m}_i; \mathbf{u}_i, \mathbf{I}), \tag{24}$$

where $\sigma(\cdot)$ is an arbitrary non-linearity. As an example, we let $\sigma(\cdot) = \texttt{Newton-Schulz}(\cdot)$, where $\texttt{Newton-Schulz}(\cdot)$ is the iterative Newton-Schulz method [66], and $\mathbf{m}(\cdot)$ be a linear layer; the resulted optimizer is equivalent to Muon optimizer [34].

Going Beyond Simple Backpropagation. As discussed earlier in Section 2.1, the pre-training process and backpropagation is a form of associative memory, where input data is mapped to the surprised caused by its predicted output $\nabla_{y_t}\mathcal{L}(W_t; x_t)$:

$$W_{t+1} = W_t - \eta_{t+1}\nabla_{W_t}\mathcal{L}(W_t; x_t) = W_t - \eta_{t+1}\nabla_{y_t}\mathcal{L}(W_t; x_t) \otimes x_t, \quad \text{where } x_t \sim \mathcal{D}_{\text{train}}, \tag{25}$$

which from the associative memory perspective is equivalent to one step of gradient descent in optimization process of:

$$\min_{W} \langle W x_t, \nabla_{y_t}\mathcal{L}(W_t; x_t)\rangle. \tag{26}$$

As we discussed in Appendix C, the above formulation cause ignoring the dependencies of data samples like x_t. To extend it to a more powerful formulation where it also consider the dependencies of data points (which is extremely important when we use optimizer in the token space as they are not independent), we use L_2 regression objective with one step of gradient descent as follows:

$$\min_{W} \|W x_t - \nabla_{y_t}\mathcal{L}(W_t; x_t)\|_2^2. \tag{27}$$

This formulation results in a new variant of gradient descent, which can be simplified as follows:

$$W_{t+1} = W_t\left(\mathbf{I} - x_t x_t^\top\right) - \eta_{t+1}\nabla_{W_t}\mathcal{L}(W_t; x_t) \tag{28}$$

$$= W_t\left(\mathbf{I} - x_t x_t^\top\right) - \eta_{t+1}\nabla_{y_t}\mathcal{L}(W_t; x_t) \otimes x_t, \qquad \text{where } x_t \sim \mathcal{D}_{\text{train}}, \tag{29}$$

Later, we use this optimizer as the internal optimizer of our HOPE architecture.

3 HOPE: A Self-Referential Learning Module with Continuum Memory

Existing architectural backbones consist of (1) a *working memory* module (e.g., attention), which is responsible to actively fuse the information across sequence length, and (2) a feed-forward layer (e.g., MLP) that fuse information across features and acts as the persistent memory or knowledge storage of pre-training phase. From the NL perspective, pre-training is the phase that the most outer level of the learning module is updated over its *limited* context flow. Accordingly, in the continual setup, such pre-training phase is also rarely updated over time, and so its corresponding knowledge storage needs to rarely be updated over time. Given this intuition, we extend the traditional view-point of long-term/short-term memory system and suggest a knowledge storage feed-forward for each level (frequency domain).

Given the definition of frequency, Continuum Memory System (CMS) is formalized as a chain of MLP blocks $\texttt{MLP}^{(f_1)}(\cdot), \ldots, \texttt{MLP}^{(f_k)}(\cdot)$, each of which associated with a chunk size of $C^{(\ell)} := \frac{\max_\ell C^{(\ell)}}{f_\ell}$ such that given input $x = \{x_1, \ldots, x_T\}$ the output of the chain is calculated as (we disregard normalizations for the sake of clarity):

$$y_t = \texttt{MLP}^{(f_k)}(\texttt{MLP}^{(f_{k-1})}(\cdots \texttt{MLP}^{(f_1)}(x_t))), \tag{30}$$

where the parameters of ℓ-th MLP block, i.e., $\boldsymbol{\theta}^{(f_\ell)}$, are updated every $C^{(\ell)}$ steps:

$$\boldsymbol{\theta}^{(f_\ell)}_{i+1} = \boldsymbol{\theta}^{(f_\ell)}_{i} - \begin{cases} \sum_{t=i-C^{(\ell)}}^{i} \eta_t^{(\ell)} f(\boldsymbol{\theta}_t^{(f_\ell)}; x_t) & \text{if } \; i \equiv 0 \pmod{C^{(\ell)}}, \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

In Appendix B.1, we discuss different variants of this formulation, including fully nested MLP layers. Here $\eta_t^{(\ell)}$ are learning rates corresponds to $\boldsymbol{\theta}^{(f_\ell)}$, and $f(\cdot)$ is the error component of an arbitrary optimizer (e.g., $\nabla\mathcal{L}(\boldsymbol{\theta}_t^{(f_\ell)}; x_t)$ in gradient descent). The conventional Transformer block [27] is a special instance of this formulation, where $k = 1$. It is notable that Equation 31 provides an important interpretation: parameters $\boldsymbol{\theta}_t^{(f_\ell)}$ are responsible for compressing their own context into the their parameters and so they are a representative of abstract knowledge of their context.

HOPE. We further present a self-referential learning module based on Titans [28] and our variant of gradient descent in Section B.1. Combining this self-referential sequence model with continuum memory system results in HOPE architecture.

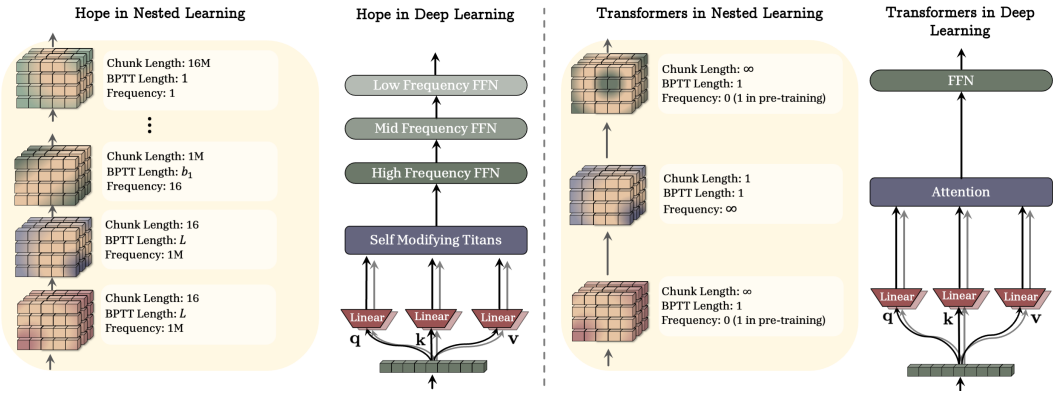

Figure 3: A comparison of Hope architectural backbone with Transformers (Normalization and potential data-dependent components are removed for the sake of clarity).

Table 1: Performance of HOPE and baselines on language modeling and common-sense reasoning tasks. Hybrid models are marked with *.

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.** ↑
HOPE (ours)	26.05	29.38	35.40	64.62	40.11	51.19	56.92	28.49	38.33	60.12	46.90
760M params / 30B tokens											
Transformer++	25.21	27.64	35.78	66.92	42.19	51.95	60.38	32.46	39.51	60.37	48.69
RetNet	26.08	24.45	34.51	67.19	41.63	52.09	63.17	32.78	38.36	57.92	48.46
DeltaNet	24.37	24.60	37.06	66.93	41.98	50.65	64.87	31.39	39.88	59.02	48.97
TTT	24.17	23.51	34.74	67.25	43.92	50.99	64.53	33.81	40.16	59.58	47.32
Samba*	20.63	22.71	39.72	69.19	47.35	52.01	66.92	33.20	38.98	61.24	51.08
Titans (LMM)	20.04	21.96	37.40	69.28	48.46	52.27	66.31	35.84	40.13	62.76	51.56
HOPE (ours)	20.53	20.47	39.02	70.13	49.21	52.70	66.89	36.05	40.71	63.29	52.26
1.3B params / 100B tokens											
Transformer++	18.53	18.32	42.60	70.02	50.23	53.51	68.83	35.10	40.66	57.09	52.25
RetNet	19.08	17.27	40.52	70.07	49.16	54.14	67.34	33.78	40.78	60.39	52.02
DeltaNet	17.71	16.88	42.46	70.72	50.93	53.35	68.47	35.66	40.22	55.29	52.14
Samba*	16.13	13.29	44.94	70.94	53.42	55.56	68.81	36.17	39.96	62.11	54.00
Titans (LMM)	15.60	11.41	49.14	73.09	56.31	59.81	72.43	40.82	42.05	60.97	56.82
HOPE (ours)	15.11	11.63	50.01	73.29	56.84	60.19	72.30	41.24	42.52	61.46	57.23

4 Experiments

For the sake of space, in the main paper, we report the results of the HOPE's evaluation on language modeling, and common-sense reasoning, tasks. However, we report an extensive set of results, including on experiments on optimizers, emergence of in-context learning, continual learning abilities of HOPE, ablation studies, long-context tasks, etc. in the appendix. Details about the experimental setups and other used datasets are in Appendix G

Language Modeling and Common-sense Reasoning. We follow recent sequence modeling studies [28, 67, 68] and report the results of HOPE and baselines with size of 340M, 760M, and 1.3B on language modeling and also commonsense reasoning downstream tasks. These results are reported in Table 1. HOPE demonstrate a very good perfomance across all the scales and benchmark tasks, outperforming both Transformers and recent modern recurrent neural networks, including Gated DeltaNet and Titans. Comparing HOPE to Titans and Gated DeltaNet, we can see that dynamically changing the key, value, and query projections based on the context as well a deep memory module can result in a model with lower perplexity and higher accuracy in benchmark results.

References

[1] Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. *arXiv preprint arXiv*.

[2] Walter Pitts. The linear theory of neuron networks: The dynamic problem. *The bulletin of mathematical biophysics*, 5:23–31, 1943.

[3] Warren S McCulloch. The brain computing machine. *Electrical Engineering*, 68(6):492–497, 1949.

[4] Warren S McCulloch and Walter Pitts. The statistical organization of nervous activity. *Biometrics*, 4(2):91–99, 1948.

[5] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[6] David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 1, 2025.

[7] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. 1998.

[8] Jonathan H. Connell and Sridhar Mahadevan. Robot learning. *Robotica*, 17(2):229–235, 1999. doi: 10.1017/S0263574799271172.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[10] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[11] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[12] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[15] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[16] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[17] 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. *arXiv preprint arXiv:2303.08774*, 2023.

[18] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *Advances in neural information processing systems*, 27, 2014.

[19] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Advances in neural information processing systems*, 29, 2016.

[20] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

[21] William Merrill, Ashish Sabharwal, and Noah A Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10: 843–856, 2022.

[22] Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Transformers, parallel computation, and logarithmic depth. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=QCZabhKQhB.

[23] 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.

[24] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[25] Juergen Schmidhuber and Sepp Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

[26] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

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

[29] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[31] Shaden Alshammari, John Hershey, Axel Feldmann, William T Freeman, and Mark Hamilton. I-con: A unifying framework for representation learning. *arXiv preprint arXiv:2504.16929*, 2025.

[32] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bklr3j0cKX.

[33] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[34] K Jordan, Y Jin, V Boza, Y Jiacheng, F Cecista, L Newhouse, and J Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024b. *URL https://kellerjordan. github. io/posts/muon*, 2024.

[35] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2018.

[36] Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M. Kakade. SOAP: Improving and stabilizing shampoo using adam for language modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=IDxZhXrpNf.

[37] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[38] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[39] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? *Advances in neural information processing systems*, 36:55565–55581, 2023.

[40] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=iaYcJKpY2B_.

[41] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *Advances in Neural Information Processing Systems*, 36: 61501–61513, 2023.

[42] Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, et al. Cartridges: Lightweight and general-purpose long context representations via self-study. *arXiv preprint arXiv:2506.06266*, 2025.

[43] hongzhou yu, Tianhao Cheng, Yingwen Wang, Wen He, Qing Wang, Ying Cheng, Yuejie Zhang, Rui Feng, and Xiaobo Zhang. FinemedLM-o1: Enhancing medical knowledge reasoning ability of LLM from supervised fine-tuning to test-time training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=7ZwuGZCopw.

[44] Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. In *Forty-second International Conference on Machine Learning*, 2024.

[45] William Beecher Scoville and Brenda Milner. Loss of recent memory after bilateral hippocampal lesions. *Journal of neurology, neurosurgery, and psychiatry*, 20(1):11, 1957.

[46] Alvaro Pascual-Leone, Amir Amedi, Felipe Fregni, and Lotfi B Merabet. The plastic human brain cortex. *Annu. Rev. Neurosci.*, 28(1):377–401, 2005.

[47] Michael V Johnston. Plasticity in the developing brain: implications for rehabilitation. *Developmental disabilities research reviews*, 15(2):94–101, 2009.

[48] Akihiro Goto, Ayaka Bota, Ken Miya, Jingbo Wang, Suzune Tsukamoto, Xinzhi Jiang, Daichi Hirai, Masanori Murayama, Tomoki Matsuda, Thomas J. McHugh, Takeharu Nagai, and Yasunori Hayashi. Stepwise synaptic plasticity events drive the early phase of memory consolidation. *Science*, 374(6569):857–863, 2021. doi: 10.1126/science.abj9195. URL https://www.science.org/doi/abs/10.1126/science.abj9195.

[49] Uwe Frey and Richard GM Morris. Synaptic tagging and long-term potentiation. *Nature*, 385 (6616):533–536, 1997.

[50] Wannan Yang, Chen Sun, Roman Huszár, Thomas Hainmueller, Kirill Kiselev, and György Buzsáki. Selection of experience for memory by hippocampal sharp wave ripples. *Science*, 383 (6690):1478–1483, 2024.

[51] Daoyun Ji and Matthew A Wilson. Coordinated memory replay in the visual cortex and hippocampus during sleep. *Nature neuroscience*, 10(1):100–107, 2007.

[52] Adrien Peyrache, Mehdi Khamassi, Karim Benchenane, Sidney I Wiener, and Francesco P Battaglia. Replay of rule-learning related neural patterns in the prefrontal cortex during sleep. *Nature neuroscience*, 12(7):919–926, 2009.

[53] David J Foster and Matthew A Wilson. Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, 440(7084):680–683, 2006.

[54] Sean PA Drummond, Gregory G Brown, J Christian Gillin, John L Stricker, Eric C Wong, and Richard B Buxton. Altered brain response to verbal learning following sleep deprivation. *Nature*, 403(6770):655–657, 2000.

[55] Seung-Schik Yoo, Peter T Hu, Ninad Gujar, Ferenc A Jolesz, and Matthew P Walker. A deficit in the ability to form new human memories without sleep. *Nature neuroscience*, 10(3):385–392, 2007.

[56] W Scott Terry. *Learning and memory: Basic principles, processes, and procedures*. Routledge, 2017.

[57] Hideyuki Okano, Tomoo Hirano, and Evan Balaban. Learning and memory. *Proceedings of the National Academy of Sciences*, 97(23):12403–12404, 2000.

[58] 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. *arXiv preprint arXiv:2504.13173*, 2025.

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

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

[61] 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. *arXiv preprint arXiv:2307.08621*, 2023.

[62] Juergen Schmidhuber. Learning to control fast-weight memories: An alternative to recurrent nets. accepted for publication in. *Neural Computation*, 1992.

[63] Imanol Schlag, Kazuki Irie, and Juergen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pages 9355–9366. PMLR, 2021.

[64] DL Prados and SC Kak. Neural network capacity using delta rule. *Electronics Letters*, 25(3): 197–199, 1989.

[65] 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. *arXiv preprint arXiv:2407.04620*, 2024.

[66] Nicholas J Higham. *Functions of matrices: theory and computation*. SIAM, 2008.

[67] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.

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

[69] 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. *arXiv preprint arXiv:2402.08132*, 2024.

[70] 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.

[71] 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.

[72] 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.

[73] Ali Behrouz, Michele Santacatterina, and Ramin Zabih. Mambamixer: Efficient selective state space models with dual token and channel selection. *arXiv preprint arXiv:2403.19888*, 2024.

[74] 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. *arXiv preprint arXiv:2404.05892*, 2024.

[75] 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. *arXiv preprint arXiv:2503.14456*, 2025.

[76] Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Increasing the expressivity of deltanet through products of householders. *arXiv preprint arXiv:2502.10297*, 2025.

[77] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[78] Juergen 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*, pages 460–463. Springer, 1993.

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

[80] Tsendsuren Munkhdalai and Hong Yu. Neural semantic encoders. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 1, page 397. NIH Public Access, 2017.

[81] Tsendsuren Munkhdalai, Alessandro Sordoni, Tong Wang, and Adam Trischler. Metalearned neural memory. *Advances in Neural Information Processing Systems*, 32, 2019.

[82] Kazuki Irie, Imanol Schlag, Robert Csordas, and Juergen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. *Advances in neural information processing systems*, 34:7703–7717, 2021.

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

[84] Kazuki Irie, Robert Csordas, and Juergen Schmidhuber. The dual form of neural networks revisited: Connecting test time predictions to training patterns via spotlights of attention. In *International Conference on Machine Learning*, pages 9639–9659. PMLR, 2022.

[85] Kazuki Irie, Imanol Schlag, Róbert Csordás, and Juergen Schmidhuber. A modern self-referential weight matrix that learns to modify itself. In *International Conference on Machine Learning*, pages 9660–9677. PMLR, 2022.

[86] Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. Can mamba learn how to learn? a comparative study on in-context learning tasks. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=GbFluKMmtE.

[87] 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.

[88] 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 Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

[89] 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*, volume 34, pages 7432–7439, 2020.

[90] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluis Marquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

[91] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[92] 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. *arXiv preprint arXiv:1803.05457*, 2018.

[93] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

[94] 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 Jill Burstein, Christy Doran, and Thamar Solorio, editors, *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)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

[95] Michael Poli, Armin W Thomas, Eric Nguyen, Pragaash Ponnusamy, Björn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Ré, et al. Mechanistic design and scaling of hybrid architectures. *arXiv preprint arXiv:2403.17844*, 2024.