What's Next for Mamba?
Towards More Expressive Recurrent Update Rules

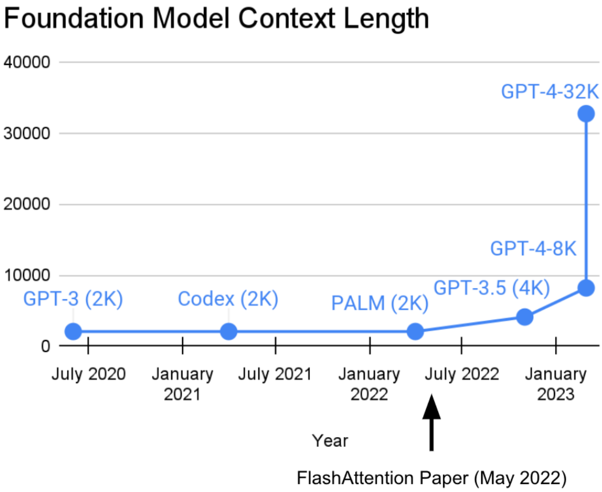
Songlin Yang

MIT CSAIL

January 2025

Introduction

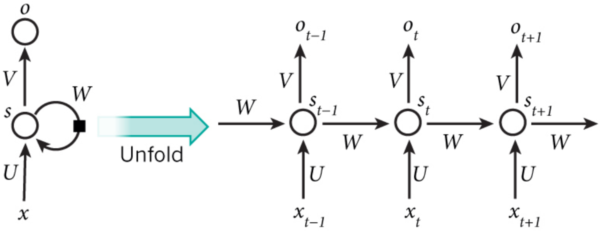
Foundation Model's Context Length grows rapidly



Issues with Transformers

- Training: quadratic time complexity
  - Expensive for long sequence modeling (e.g., video or DNA modeling)
- Inference: linear memory complexity
  - Requires storing KV cache for each token
  - High memory burden.

Revisiting RNNs



- Training: linear complexity, however, traditional RNNs are not parallelizable.
- Inference: constant memory

Modern linear recurrent models

Use linear recurrence for parallel training

- Gated linear RNNs (HGRN, Griffin, ...)
- State-space models (S4, Mamba, ...)
- Linear attention (RetNet, GLA, xLSTM, DeltaNet, ...)

Modern linear recurrent models

Use linear recurrence for parallel training

- Gated linear RNNs (HGRN, Griffin, ...)
- State-space models (S4, Mamba, ...)
- **Linear attention (RetNet, GLA, xLSTM, DeltaNet, ...)**

Mamba2 is more similar to linear attention than state-space models!!

Hybrid linear and softmax attention can achieve GPT-4o level performance

MiniMax-01 (MiniMax et al. 2025) used

- Hybrid attention: **7/8** linear attention layers + **1/8** softmax attention layer
- Lightning attention (Qin et al. 2024b): simple linear attention with data-independent decay

Linear attention background

Linear attention = standard attention - softmax

Softmax attention:

$$\text{Parallel training}: \quad \mathbf{O} = \mathrm{softmax}(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V} \quad \in \mathbb{R}^{L\times d}$$

$$\text{Iterative inference}: \quad \mathbf{o_t} = \sum_{j=1}^{t} \frac{\exp(\mathbf{q}_t^\top \mathbf{k}_j)}{\sum_{l=1}^{t} \exp(\mathbf{q}_t^\top \mathbf{k}_l)} \mathbf{v}_j \quad \in \mathbb{R}^{d}$$

where $\mathbf{M} \in \mathbb{R}^{L\times L}$ is the casual mask:

$$\mathbf{M}_{i,j} = \begin{cases} -\infty & \text{if } j > i \\ 1 & \text{if } j \leq i \end{cases}$$

Linear attention = standard attention - softmax

Linear attention (Katharopoulos et al. 2020):

$$\text{Parallel training}: \quad \mathbf{O} = \sout{\text{softmax}}(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V} \quad \in \mathbb{R}^{L \times d}$$

$$\text{Iterative inference}: \quad \mathbf{o_t} = \sum_{j=1}^{t} \frac{\sout{\exp}(\mathbf{q}_t^\top \mathbf{k}_j)}{\sout{\sum_{l=1}^{t} \exp(\mathbf{q}_t^\top \mathbf{k}_l)}} \mathbf{v}_j \quad \in \mathbb{R}^{d}$$

where the denominator is harmful for linear attention's training stability and performance (Qin et al. 2022). Therefore, nearly all recent linear attention models remove this normalization term.

Linear attention = standard attention - softmax

Linear attention (Katharopoulos et al. 2020):

$$\text{Parallel training}: \quad \mathbf{O} = \cancel{\text{softmax}}(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V} \quad \in \mathbb{R}^{L\times d}$$

$$\text{Iterative inference}: \quad \mathbf{o_t} = \sum_{j=1}^{t} \frac{\cancel{\exp}(\mathbf{q}_t^\top \mathbf{k}_j)}{\cancel{\sum_{l=1}^{t} \exp(\mathbf{q}_t^\top \mathbf{k}_l)}} \mathbf{v}_j \quad \in \mathbb{R}^{d}$$

We abuse the notation **M** to denote the causal mask for both softmax and linear attention. Here we have:

$$\mathbf{M}_{i,j} = \begin{cases} 0 & \text{if } j > i \\ 1 & \text{if } j \le i \end{cases}$$

Linear attention = Linear RNN + matrix-valued hidden states

$$
\begin{aligned}
\mathbf{o_t} &= \sum_{j=1}^{t} (\mathbf{q}_t^\top \mathbf{k}_j)\mathbf{v}_j \\
&= \sum_{j=1}^{t} \mathbf{v}_j(\mathbf{k}_j^\top \mathbf{q}_t) \quad \mathbf{k}_j^\top \mathbf{q}_t = \mathbf{q}_t^\top \mathbf{k}_j \in \mathbb{R} \\
&= \underbrace{\left(\sum_{j=1}^{t} \mathbf{v}_j \mathbf{k}_j^\top\right)}_{\mathbf{S}_t \in \mathbb{R}^{d \times d}} \mathbf{q}_t \qquad \text{By associativity}
\end{aligned}
$$

Linear attention = Linear RNN + matrix-valued hidden states

Let $\mathbf{S}_t = \sum_{j=1}^{t} \mathbf{v}_j \mathbf{k}_j^\top \in \mathbb{R}^{d \times d}$ be the matrix-valued hidden state, then:

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \quad \in \mathbb{R}^{d \times d}$$
$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \quad \in \mathbb{R}^{d}$$

- Linear attention implements **elementwise linear recurrence**.
- Linear attention has a **matrix-valued hidden state**, significantly increasing the state size.

Challenges in linear attention training: the parallel form

$$\mathbf{O} = (\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V} \in \mathbb{R}^{L \times d}$$

- Still quadratic in sequence length.

Challenges in linear attention training: the recurrent form

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \quad \in \mathbb{R}^{d \times d}$$
$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \quad \in \mathbb{R}^{d}$$

- Sequential computation limits parallelization opportunities
- Poor GPU utilization due to lack of matrix-multiply operations (even with parallel scan algorithms)

Challenges in linear attention training: the recurrent form

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \quad \in \mathbb{R}^{d \times d}$$
$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \quad \in \mathbb{R}^{d}$$

- Sequential computation limits parallelization opportunities
- Poor GPU utilization due to lack of matrix-multiply operations (even with parallel scan algorithms)

Hardware-efficient training with chunkwise parallel form

- Sequence of length L divided into L/C chunks of size C
- Compute only the **last hidden state** of each chunk.
- Compute the output from two parts:
  - Historical context: using **recurrent** form
  - Local context: using **parallel** form

Hardware-efficient training with chunkwise parallel form

- Sequence of length L divided into L/C chunks of size C
- Compute only the last hidden state of each chunk.
- Compute the output from two parts:
  - Historical context: using recurrent form
  - Local context: using parallel form
- When $C = 1$, it reduces to recurrent form; when $C = L$, it reduces to parallel form.
- Chunkwise form is **NOT an approximation**, it computes the exact same output.

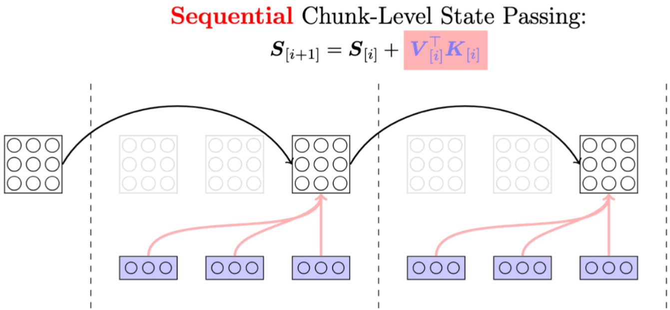
Notation:

$$\mathbf{S}_{[i]} := \mathbf{S}_{iC} \in \mathbb{R}^{d \times d} \qquad \text{(Chunk-level hidden state)}$$

$$\square_{[i]} = \square_{iC+1:(i+1)C} \in \mathbb{R}^{C \times d} \qquad \text{(Matrix block for chunk } i\text{)}$$

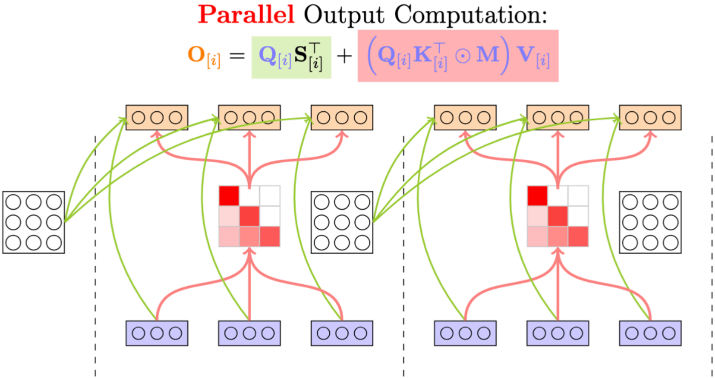
$$\text{for } \square \in \{\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}\}$$

Chunkwise parallel form: hidden state update



$$\mathbf{S}_{[t+1]} = \underbrace{\mathbf{S}_{[t]}}_{\mathbb{R}^{d\times d}} + \underbrace{\mathbf{V}_{[t]}^\top}_{\mathbb{R}^{d\times C}} \underbrace{\mathbf{K}_{[t]}}_{\mathbb{R}^{C\times d}} \quad \in \mathbb{R}^{d\times d} \qquad \text{(Matrix Form)}$$

Chunkwise parallel form: parallel output computation

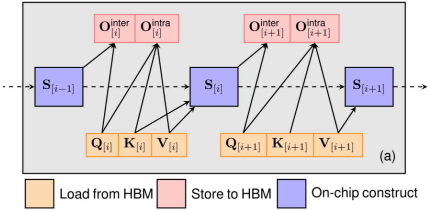


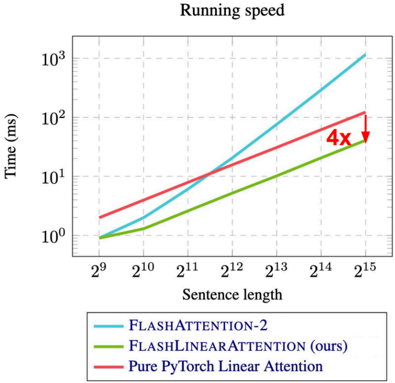
$$\mathbf{O}_{[t]} = \underbrace{\overbrace{\mathbf{Q}_{[t]}}^{\mathbb{R}^{C\times d}}\overbrace{\mathbf{S}_{[t]}^\top}^{\mathbb{R}^{d\times d}}}_{\text{inter-chunk:}\mathbf{O}_{[t]}^{\text{inter}}} + \underbrace{\overbrace{(\mathbf{Q}_{[t]}\mathbf{K}_{[t]}^\top \odot \mathbf{M})}^{\mathbb{R}^{C\times C}}\overbrace{\mathbf{V}_{[t]}}^{\mathbb{R}^{C\times d}}}_{\text{intra-chunk:}\mathbf{O}_{[t]}^{\text{intra}}} \in \mathbb{R}^{C\times d} \qquad \text{(Matrix Form)}$$

Chunkwise parallel form

- Total complexity: $\mathcal{O}(Ld^2 + LdC)$, subquadratic in sequence length when C is set small.
- C is set to $\{64, 128, 256\}$ in practice.
- Can be extended to linear attention with decay and delta rule (which we will discuss later).
- The de facto standard for training modern linear attention models (e.g., Mamba2, Based, GLA, DeltaNet, Lightning Attention, mLSTM $\cdots$)

Flash linear attention





I/O optimization significantly improves the wall-clock time.

Flash linear attention library

The Flash Linear Attention library provides hardware-efficient implementation of various linear attention models.

- RetNet, GLA, Based, HGRN2, RWKV6, GSA, Mamba2, DeltaNet, Gated DeltaNet, RWKV7 ...

Linear attention is not enough

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t\mathbf{k}_t^\top \qquad \in \mathbb{R}^{d\times d}$$

$$\mathbf{o}_t = \mathbf{S}_t\mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

- **Instability:** the hidden state value could explode due to cumulative sum without decay
- **Poor performance:** vanilla linear attention models significantly underperform Transformers in language modeling perplexity

A simple fix: linear attention with data-independent decay

$$
\begin{aligned}
\mathbf{S}_t &= \gamma \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top && \in \mathbb{R}^{d \times d} \\
\mathbf{o}_t &= \mathbf{S}_t \mathbf{q}_t && \in \mathbb{R}^{d}
\end{aligned}
$$

- γ is a constant exponential decay factor $0 < \gamma < 1$.
- Works well in practice: RetNet (Sun et al. 2023), Lightning Attention (Qin et al. 2024b)
- Lacking selectivity: a potential issue.

A simple fix: linear attention with data-dependent decay

$$\mathbf{S}_t = \gamma_t \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$
$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

- $\gamma_t \in (0, 1)$ is a data-dependent decay term
- Enables dynamic control of memory retention/forgetting based on input data.
- Examples: Mamba2 (Dao and Gu 2024), mLSTM (Beck et al. 2024), Gated Retention (Sun et al. 2024b).

The parallel form for linear attention with decay

$$
\begin{aligned}
\mathbf{S}_t &= \gamma_t \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top && \in \mathbb{R}^{d \times d} \\
\mathbf{o}_t &= \mathbf{S}_t \mathbf{q}_t && \in \mathbb{R}^{d}
\end{aligned}
$$

Linear attention with decay has the following parallel form:

$$
\mathbf{O} = (\mathbf{Q}\mathbf{K}^\top \odot \mathbf{D})\mathbf{V} \in \mathbb{R}^{L \times d}
$$

$$
\mathbf{D}_{i,j} = \begin{cases} \prod_{m=i+1}^{j} \gamma_m & \text{if } i < j \\ 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}
$$

Here, the equivalence of the recurrent and the parallel form is also known as **state space duality** in Mamba2 (Dao and Gu 2024).

The chunkwise parallel form for linear attention with decay

$$\mathbf{S}_t = \gamma_t \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \quad \in \mathbb{R}^{d\times d} \qquad \mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \quad \in \mathbb{R}^d$$

Linear attention with decay has the following chunkwise form:

$$\mathbf{S}_{[t]} = \beta_{tC} \mathbf{S}_{[t-1]} + (\mathbf{V}_{[t]} \odot \frac{\beta_{tC}}{\beta_{[t]}}[:, \text{None}])^\top \mathbf{K}_{[t]} \quad \in \mathbb{R}^{d\times d}$$

$$\mathbf{O}_{[t]} = \underbrace{(\mathbf{Q}_{[t]} \mathbf{S}_{[t]}^\top) \odot \beta_{[t]}[:, \text{None}]}_{\text{inter-chunk}} + \underbrace{(\mathbf{Q}_{[t]} \mathbf{K}_{[t]}^\top \odot \mathbf{D}_{[t]}) \mathbf{V}_{[t]}}_{\text{intra-chunk}} \quad \in \mathbb{R}^{C\times d}$$

where

- $\beta_{[t]} \in \mathbb{R}^C$ represents the cumulative decay values within chunk t, where the i-th element is $(\beta_{[t]})_i = \beta_{tC+i} = \prod_{m=tC+1}^{tC+i} \gamma_m \in \mathbb{R}$
- $(\mathbf{D}_{[t]})_{i,j} = \begin{cases} \prod_{m=tC+i+1}^{tC+j} \gamma_m & \text{if } i < j \\ 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad \in \mathbb{R}^{C\times C}$
- Reminder: $\mathbf{S}_{[t]} := \mathbf{S}_{tC} \in \mathbb{R}^{d\times d}$. $\square_{[t]} := \square_{tC+1:(t+1)C} \in \mathbb{R}^{C\times d}$ for $\square \in \{\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}\}$.

The chunkwise parallel form for linear attention with decay

$$\mathbf{S}_t = \gamma_t \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \quad \in \mathbb{R}^{d \times d} \qquad \mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \quad \in \mathbb{R}^{d}$$

Linear attention with decay has the following chunkwise form:

$$\mathbf{S}_{[t]} = \beta_{Ct} \mathbf{S}_{[t-1]} + \left(\mathbf{V}_{[t]} \odot \frac{\beta_{tC}}{\beta_{[t]}}[:, \text{None}]\right)^\top \mathbf{K}_{[t]} \quad \in \mathbb{R}^{d \times d}$$

$$\mathbf{O}_{[t]} = \underbrace{(\mathbf{Q}_{[t]} \mathbf{S}_{[t]}^\top) \odot \beta_{[t]}[:, \text{None}]}_{\text{inter-chunk}} + \underbrace{(\mathbf{Q}_{[t]} \mathbf{K}_{[t]}^\top \odot \mathbf{D}_{[t]}) \mathbf{V}_{[t]}}_{\text{intra-chunk}} \quad \in \mathbb{R}^{C \times d}$$

- Preserves matrix-multiply structure with minimal overhead when incorporating decay
- As fast as vanilla linear attention's chunkwise form
- Equivalent to Mamba2's **state space duality (SSD) algorithm** (Dao and Gu 2024)

Linear attention with more fine-grained decay

$$\mathbf{S}_t = \mathbf{G}_t \odot \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

- $\mathbf{G}_t \in \mathbb{R}^{d \times d}$ is a fine-grained data-dependent gate matrix.
- $\mathbf{G}_t = \boldsymbol{\beta}_t \boldsymbol{\alpha}_t^\top$ enables rewriting recurrence into matrix-multiply form (Yang et al. 2023), where $\boldsymbol{\beta}_t, \boldsymbol{\alpha}_t \in \mathbb{R}^d$ are learnable data-dependent vectors.
- $\mathbf{G}_t = \exp(-(\boldsymbol{\Delta}_t \mathbf{1}^\top) \odot \exp(\mathbf{A}))$ in Mamba1 (Gu and Dao 2023):
  - $\mathbf{A} \in \mathbb{R}^{d \times d}$ is data-independent, $\boldsymbol{\Delta}_t \in \mathbb{R}^d$ is data-dependent.
  - Breaks down the outer product form and therefore lacks the matrix-multiply form.
  - Difficult to scale up the recurrent state size.

Linear attention with more fine-grained decay

$$\mathbf{S}_t = \mathbf{G}_t \odot \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$
$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

- $\mathbf{G}_t \in \mathbb{R}^{d \times d}$ is a fine-grained data-dependent gate matrix.
- $\mathbf{G}_t = \boldsymbol{\beta}_t \boldsymbol{\alpha}_t^\top$ enables rewriting recurrence into matrix-multiply form (Yang et al. 2023), where $\boldsymbol{\beta}_t, \boldsymbol{\alpha}_t \in \mathbb{R}^d$ are learnable data-dependent vectors.
- $\mathbf{G}_t = \exp(-(\boldsymbol{\Delta}_t \mathbf{1}^\top) \odot \exp(\mathbf{A}))$ in Mamba1 (Gu and Dao 2023):
  - $\mathbf{A} \in \mathbb{R}^{d \times d}$ is data-independent, $\boldsymbol{\Delta}_t \in \mathbb{R}^d$ is data-dependent.
  - Breaks down the outer product form and therefore lacks the matrix-multiply form.
  - Difficult to scale up the recurrent state size.

Linear attention with more fine-grained decay

$$\mathbf{S}_t = \mathbf{G}_t \odot \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

- Full form: $\mathbf{G}_t = \boldsymbol{\beta}_t \boldsymbol{\alpha}_t^\top$
  - Examples: Decaying fast weight (Mao 2022)
  - Hardware-efficient training with chunkwise parallel form (Yang et al. 2023).
- A simpler choice: $\mathbf{G}_t = \mathbf{1}\boldsymbol{\alpha}_t^\top$ by setting $\boldsymbol{\beta}_t = \mathbf{1}$
  - Faster than the full form case, but still slower than linear attention with scalar-valued decay (e.g., Lightning Attention, Mamba2).
  - Examples: GLA (Yang et al. 2023), RWKV6 (Peng et al. 2024), MetaLA (Chou et al. 2024), HGRN2 (Qin et al. 2024a), GSA (Zhang et al. 2024)

Towards more expressive update rule

Linear attention: a fast weight programming perspective

The hidden state matrix $\mathbf{S}_t$ is a fast weight matrix that is updated at each timestep:

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top$$

The fast weight matrix is used to map inputs $\mathbf{q}_t$ into outputs $\mathbf{o}_t$:

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t$$

> *"Fast weights provide a neurally plausible way of implementing the type of temporary storage that is required by working memory, while slow weights capture more permanent associations learned over many experiences." – Geoffrey Hinton*

The choice of update rule

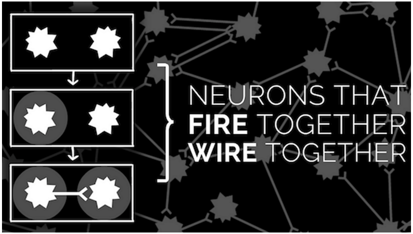


Figure: The principle of Hebbian learning.

- Hebbian update rule: $\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top$
- Delta rule: $\mathbf{S}_t = \mathbf{S}_{t-1} - \beta_t \left(\mathbf{S}_{t-1} \mathbf{k}_t - \mathbf{v}_t \right) \mathbf{k}_t^\top$
- ...

Both Hebbian and delta update rules can be regarded as optimizing online learning objective via single step of SGD.

Linear attention optimizes a negative linear inner product loss via SGD

The objective predicts the target value $\mathbf{v}_t$ by transforming the key $\mathbf{k}_t$ with $\mathbf{S}$.

$$\mathcal{L}_t(\mathbf{S}) = -\langle \mathbf{S}\mathbf{k}_t, \mathbf{v}_t \rangle$$

Performing a single step of SGD:

$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_{t-1} - \beta_t \nabla \mathcal{L}_t(\mathbf{S}_{t-1}) \\ &= \mathbf{S}_{t-1} + \beta_t \mathbf{v}_t \mathbf{k}_t^\top \end{aligned}$$

- Learning rate $\beta_t = 1$ recovers vanilla linear attention.
- Mamba2's update rule $\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top$ can be interpreted as online SGD with weight decay α_t.

DeltaNet optimizes a regression loss via SGD

Online regression loss is better for predicting $\mathbf{v}_t$ from $\mathbf{k}_t$ and $\mathbf{S}_{t-1}$.

$$\mathcal{L}_t(\mathbf{S}) = \frac{1}{2}\|\mathbf{S}\mathbf{k}_t - \mathbf{v}_t\|^2$$

Performing a single step of SGD:

$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_{t-1} - \beta_t \nabla \mathcal{L}_t(\mathbf{S}_{t-1}) \\ &= \mathbf{S}_{t-1} - \beta_t \left(\mathbf{S}_{t-1}\mathbf{k}_t - \mathbf{v}_t\right) \mathbf{k}_t^\top \end{aligned}$$

- When $\beta_t \in (0, 1)$, the DeltaNet update rule (Schlag, Irie, and Schmidhuber 2021; Yang et al. 2024) is recovered.

DeltaNet performs better on in-context associative recall: key intuitions

What is associative recall?

In psychology, associative memory is the ability to learn and remember relationships between unrelated items, such as remembering someone's name when seeing their face. This cognitive mechanism allows us to form and retrieve connections between distinct pieces of information. - Wikipedia

DeltaNet's online regression loss directly optimizes the model's ability to predict $\mathbf{v}_i$ from their corresponding key vectors $\mathbf{k}_i$ at each step, enhancing key-value associative recall (Liu et al. 2024).

DeltaNet performs better on in-context associative recall: MQAR results

Multi-Query Associative Recall (MQAR, Arora et al. 2023)

A synthetic benchmark for testing in-context associative recall. **Example:**

- Given key-value pairs: “A 4 B 3 C 6 F 1 E 2”
- Query: “A ? C ? F ? E ? B ?”
- Expected output: “4, 6, 1, 2, 3”

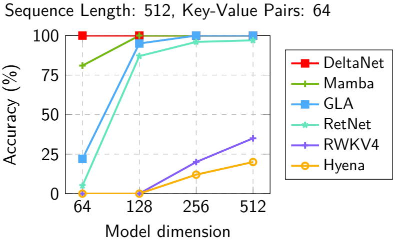


Figure: Accuracy (%) on MQAR. DeltaNet achieves the perfect recall.

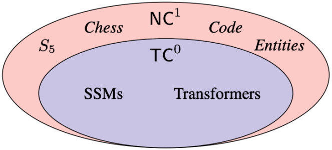
DeltaNet performs better on in-context associative recall: MAD results

MAD (Poli et al. 2024) serves as a more comprehensive benchmark suite than MQAR for evaluating in-context associative recall and learning.

Model	**Compress**	**Fuzzy Recall**	**In-Context Recall**	**Memorize**	**Noisy Recall**	**Selective Copy**	**Average**
Transformer	51.6	29.8	94.1	85.2	86.8	99.6	74.5
Hyena	45.2	7.9	81.7	89.5	78.8	93.1	66.0
Multihead Hyena	44.8	14.4	99.0	89.4	98.6	93.0	73.2
Mamba	52.7	6.7	90.4	89.5	90.1	86.3	69.3
GLA	38.8	6.9	80.8	63.3	81.6	88.6	60.0
DeltaNet	42.2	**35.7**	**100**	52.8	**100**	**100**	71.8

Table: MAD benchmark results. DeltaNet achieves the best performance in in-context associative recall and copy tasks, however, it somehow underperforms in memorization and compression tasks.

Transformers and SSMs fall under TC^0



Merrill, Petty, and Sabharwal 2024 identified two key approaches for designing RNNs with computational power beyond TC^0:

- **Nonlinear Recurrence**
  - \+ Achieves expressiveness beyond TC^0
  - \- Inherently sequential, not parallelizable
- **Linear Recurrence with Data-Dependent Nondiagonal Transition Matrices**
  - \+ Theoretically parallelizable
  - \- Could be computationally expensive with dense unstructured transition matrices.

DeltaNet has more expressive transition matrix

$$\begin{aligned}\mathbf{S}_t &= \mathbf{S}_{t-1} - \beta_t \nabla \mathcal{L}_t(\mathbf{S}_{t-1}) \\ &= \mathbf{S}_{t-1} - \beta_t \left(\mathbf{S}_{t-1}\mathbf{k}_t - \mathbf{v}_t\right) \mathbf{k}_t^\top \\ &= \mathbf{S}_{t-1}\left(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top\right) + \beta_t \mathbf{v}_t \mathbf{k}_t^\top\end{aligned}$$

Generalized Householder (GH) transition matrix: $\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top$

- DeltaNet's GH transition matrix is both **data-dependent** and **non-diagonal**.
- Strictly more expressive than Mamba2 when GH has negative eigenvalues - this allows DeltaNet to compute functions beyond the TC^0 complexity class (Grazzi et al. 2024; Merrill, Petty, and Sabharwal 2024)
- The structured GH matrix form enables efficient chunkwise training (Yang et al. 2024)

DeltaNet is strictly more expressive than SSMs

$$\mathbf{S}_t = \mathbf{S}_{t-1} \underbrace{(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)}_{\text{GH transition}} + \beta_t \mathbf{v}_t \mathbf{k}_t^\top = \sum_{i=1}^{t} \left(\beta_i \mathbf{v}_i \mathbf{k}_i^t \underbrace{\prod_{j=i+1}^{t} (\mathbf{I} - \beta_j \mathbf{k}_j \mathbf{k}_j^\top)}_{\text{cumulative GH products}} \right)$$

Key Properties:

- **Expressiveness**: When allowing negative eigenvalues in GH matrices (Grazzi et al. 2024), the cumulative products of GH matrices can represent *any* matrix with Euclidean norm < 1.
- **Complexity Class**: Cumulative products of general matrices cannot be computed in TC^0 (Mereghetti and Palano 2000).
- **Conclusion**: DeltaNet with negative eigenvalues has expressiveness beyond TC^0, strictly exceeding SSMs and Transformer.

DeltaNet with negative eigenvalue has better state tracking capability than Transformer and Mamba

	Parity	Mod. Arithm. (w/o brackets)	Mod. Arithm. w/ brackets)
Transformer	0.022	0.031	0.025
mLSTM	0.087 (0.04)	0.040 (0.04)	0.034 (0.03)
sLSTM	**1.000** (1.00)	**0.787** (1.00)	**0.173** (0.57)
Mamba $[0, 1]$	0.000	0.095	0.092
Mamba $[-1, 1]$	**1.000**	**0.241**	**0.136**
DeltaNet $[0, 1]$	0.017	0.314	0.137
DeltaNet $[-1, 1]$	**1.000**	**0.971**	**0.200**

Figure: This table is from Grazzi et al. 2024. DeltaNet has a strong state tracking capability in parity checking and modular arithmetic.

Issues with DeltaNet

Despite strong performance on synthetic benchmarks like MQAR and MAD, DeltaNet underperforms on real-world language modeling tasks compared to models like Mamba2

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.
Mamba	17.92	15.06	43.98	71.32	52.91	52.95	69.52	35.40	37.76	61.13	53.12
Mamba2	16.56	12.56	45.66	71.87	55.67	55.24	72.47	37.88	40.20	60.13	54.89
DeltaNet	17.71	16.88	42.46	70.72	50.93	53.35	68.47	35.66	40.22	55.29	52.14

Table: Performance comparison on language modeling and zero-shot common-sense reasoning for 1.3B parameter models that are trained for 100B tokens.

Decay is crucial for forgetting irrelevant information!

Gated DeltaNet (Yang, Kautz, and Hatamizadeh 2024)

Gated DeltaNet combines the delta update rule in DeltaNet and the gated update rule in Mamba2:

$$\mathbf{S}_t = \mathbf{S}_{t-1} \left(\alpha_t (\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top) \right) + \beta_t \mathbf{v}_t \mathbf{k}_t^\top$$

- $\alpha_t \in (0, 1)$ is parameterized the same as Mamba2.
- When $\alpha_t = 1$, Gated DeltaNet is equivalent to DeltaNet.
- When $\alpha_t = 0$, Gated DeltaNet clears the entire memory.
- Gated DeltaNet can be interpreted as optimizing the online regression loss with weight decay.

Case study: Single Needle In a Haystack (S-NIAH)

S-NIAH is a benchmark suite from RULER (Hsieh et al. 2024) for testing in-context associative recall capabilities through three increasingly challenging subtasks.

Task	**Configurations**		
	Subtask-1	**Subtask-2**	**Subtask-3**
Single NIAH	type_key = word type_value = number type_haystack = repeat ~passkey retrieval	type_key = word type_value = number type_haystack = essay ~vanilla NIAH	type_key = word type_value = uuid type_haystack = essay

Table: Configurations for Single NIAH Task

Case study: Single Needle In a Haystack (S-NIAH)

S-NIAH-1: A pass-key retrieval task with synthetic context

Context:

A special magic number is hidden within a long text of repeated sentences. Make sure to memorize it. I will quiz you about the number afterwards. The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. [....] One of the special magic numbers for flaky-celebrity is: 1538552. The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. [....]

Query: "What is the special magic number for flaky-celebrity?"
Expected answer: "1538552"

Case study: Single Needle In a Haystack (S-NIAH)

	S-NIAH-1 (pass-key retrieval)			
Model	1K	2K	4K	8K
DeltaNet	97.4	96.8	**99.0**	**98.8**
Mamba2	**99.2**	**98.8**	65.4	30.4
Gated DeltaNet	98.4	88.4	91.4	91.8

Decay hurts memory retention!

- Mamba2 significantly degrades with longer sequences
- DeltaNet maintains consistent performance
- Gated DeltaNet shows slight degradation

Case study: Single Needle In a Haystack (S-NIAH)

S-NIAH-2: number in a haystack

Context:

A special magic number is hidden within the following text. Make sure to memorize it. I will quiz you about the number afterwards.
What hard liquor, cigarettes, heroin, and crack have in common is that they're all more concentrated forms of less addictive predecessors. Most if not all the things we describe as addictive are. [....] One of the special magic numbers for vague-ecology is: 6440561. And the scary thing is, the process that created them is accelerating. We wouldn't want to stop it. It's the same process that cures diseases: technological progress. Technological progress means making things do more of what we want. When the thing we want is something we want to want, we consider technological progress good [....]
Query: "What is the special magic number for vague-ecology?"
Expected answer: "6440561"

- S-NIAH-2 is more challenging than S-NIAH-1, as the context is drawn from real-world essays (Paul Graham Essays) rather than synthetic text.
- Success on this task requires models to effectively filter out irrelevant information while retaining the key number.

Case study: Single Needle In a Haystack (S-NIAH)

	S-NIAH-2 (number in haystack)			
Model	1K	2K	4K	8K
DeltaNet	98.4	45.6	18.6	14.4
Mamba2	99.4	98.8	56.2	17.0
Gated DeltaNet	**100.0**	**99.8**	**92.2**	**29.6**

Data-dependent decay helps filter out irrelevant information!

- DeltaNet's performance drops significantly due to lack of decay mechanism.
- Mamba2 shows comparable performance to S-NIAH-1 task.
- Gated DeltaNet demonstrates superior performance in S-NIAH-2.

Case study: Single Needle In a Haystack (S-NIAH)

S-NIAH-3: uuid in a haystack

Context:

A special magic uuid is hidden within the following text. Make sure to memorize it. I will quiz you about the uuid afterwards.
What hard liquor, cigarettes, heroin, and crack have in common is that they're all more concentrated forms of less addictive predecessors. Most if not all the things we describe as addictive are. [....] One of the special magic uuid for vague-ecology is: 8a14be62-295b-4715-8333-e8615fb8d16c. And the scary thing is, the process that created them is accelerating. We wouldn't want to stop it. It's the same process that cures diseases: technological progress. Technological progress means making things do more of what we want. When the thing we want is something we want to want, we consider technological progress good [....]
Query: "What is the special magic uuid for vague-ecology?"
Expected answer: "8a14be62-295b-4715-8333-e8615fb8d16c"

- S-NIAH-3 is more challenging than S-NIAH-2 as the value is a uuid rather than a number.

Case study: Single Needle In a Haystack (S-NIAH)

	S-NIAH-3 (uuid in haystack)		
Model	1K	2K	4K
DeltaNet	85.2	47.0	22.4
Mamba2	64.4	47.6	4.6
Gated DeltaNet	**86.6**	**84.2**	**27.6**

Delta rule helps memorize more complex patterns!

- Mamba2's performance drops significantly due to the lack of delta rule.
- DeltaNet's performance is similar to S-NIAH-2.
- Gated DeltaNet achieves the best performance in S-NIAH-3.

Gated DeltaNet and hybrid models

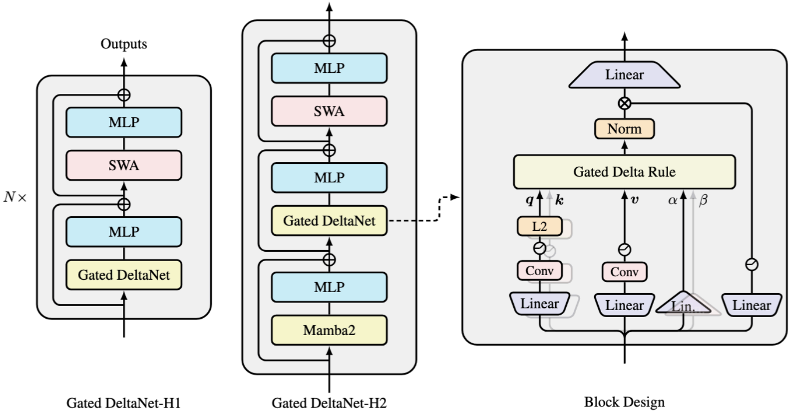


Figure: Gated DeltaNet and hybrid blocks. SWA stands for Sliding Window Attention.

Zero-shot commonsense reasoning performance

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.
Recurrent models											
RetNet	19.08	17.27	40.52	70.07	49.16	54.14	67.34	33.78	40.78	60.39	52.02
HGRN2	19.10	17.69	39.54	70.45	49.53	52.80	69.40	35.32	40.63	56.66	51.79
Mamba	17.92	15.06	43.98	71.32	52.91	52.95	69.52	35.40	37.76	61.13	53.12
Mamba2	16.56	12.56	45.66	71.87	55.67	55.24	**72.47**	37.88	40.20	60.13	54.89
DeltaNet	17.71	16.88	42.46	70.72	50.93	53.35	68.47	35.66	40.22	55.29	52.14
Gated DeltaNet	**16.42**	**12.17**	**46.65**	**72.25**	**55.76**	**57.45**	**71.21**	**38.39**	**40.63**	60.24	**55.32**
Attention or hybrid models											
Transformer++	18.53	18.32	42.60	70.02	50.23	53.51	68.83	35.10	40.66	57.09	52.25
Samba	16.13	13.29	44.94	70.94	53.42	55.56	68.81	36.17	39.96	62.11	54.00
Gated DeltaNet-H1	16.07	**12.12**	47.73	**72.57**	56.53	**58.40**	71.75	**40.10**	41.40	**63.21**	**56.40**
Gated DeltaNet-H2	**15.91**	12.55	**48.76**	72.19	**56.88**	57.77	71.33	39.07	**41.91**	61.55	56.18

Table: Performance comparison on language modeling and zero-shot common-sense reasoning for 1.3B parameter models that are trained for 100B tokens.

Zero-shot long context understanding performance

Model	Single-Doc QA			Multi-Doc QA			Summarization			Few-shot			Code		Avg
	NQA	QQA	MFQ	HQA	2WM	Mus	GvR	QMS	MNs	TRC	TQA	SSM	LCC	RBP	
Recurrent models															
RetNet	12.1	10.7	19.1	10.7	**18.0**	5.8	4.8	15.8	7.9	19.0	18.0	12.8	14.1	17.9	13.2
HGRN2	10.7	12.1	19.1	11.3	15.7	6.0	5.2	15.1	**9.2**	16.0	15.8	10.3	18.6	20.8	13.5
Mamba	13.0	10.1	20.4	10.1	16.7	6.0	7.2	15.9	8.4	23.1	21.9	11.2	17.9	19.0	14.6
DeltaNet	12.9	10.8	21.5	10.9	13.2	5.1	6.5	13.5	7.2	15.5	23.3	11.6	17.6	20.3	13.6
Mamba2	11.1	11.3	18.6	11.8	15.1	**6.7**	6.7	14.5	7.4	13.0	**23.6**	8.4	17.9	20.6	13.5
Gated DeltaNet	**14.1**	**14.0**	**23.3**	**13.7**	14.4	5.8	**7.5**	**16.4**	7.9	**30.0**	22.4	**23.0**	**18.7**	**22.1**	**16.6**
Attention or hyrbid models															
Transformer++	11.8	9.3	10.0	10.9	4.2	6.1	7.4	15.8	6.6	16.9	13.5	3.9	17.2	18.7	11.0
Samba	12.5	12.9	25.4	11.2	19.7	6.8	9.1	15.7	11.0	20.0	22.7	22.8	18.1	21.1	15.9
Gated DeltaNet-H1	**14.5**	12.3	26.6	12.6	**23.6**	6.1	9.1	16.1	12.8	33.5	**23.9**	26.8	15.5	19.2	17.8
Gated DeltaNet-H2	12.7	**13.0**	**27.1**	**12.7**	20.6	**7.5**	**10.4**	**16.2**	**13.0**	**40.5**	22.7	**27.9**	**19.9**	**22.1**	**18.4**

Table: Accuracy on 14 tasks from LongBench (Bai et al. 2023): Narrative QA, QasperQA, MultiField QA, HotpotQA, 2WikiMulti QA, Musique, GovReport, QMSum, MultiNews, TRec, Trivia QA, SamSum, LCC, and RepoBench-P by order.

- Transformer performs poorly due to limited length extrapolation without long sequence post-training.

Parallelizing DeltaNet (Yang et al. 2024)

$$\mathbf{S}_t = \mathbf{S}_{t-1}\left(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top\right) + \beta_t \mathbf{v}_t \mathbf{k}_t^\top$$

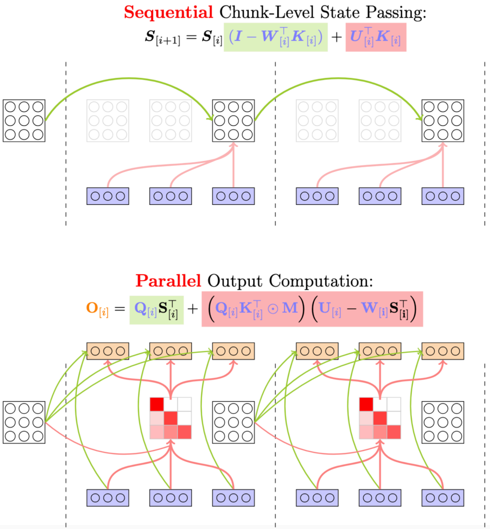
$$= \sum_{i=1}^{t} \left(\beta_i \mathbf{v}_i \mathbf{k}_i^t \underbrace{\prod_{j=i+1}^{t} (\mathbf{I} - \beta_j \mathbf{k}_j \mathbf{k}_j^\top)}_{\mathbf{P}_j^t} \right)$$

$\mathbf{S}_t$ and $\mathbf{P}_t := \mathbf{P}_1^t$ can be computed efficiently via the classical WY representation (Bischof and Loan 1985):

$$\mathbf{P}_t = \mathbf{I} - \sum_{i=1}^{t} \mathbf{w}_i \mathbf{k}_i^\top, \qquad \mathbf{w}_t = \beta_t \left(\mathbf{k}_t - \sum_{i=1}^{t-1} \mathbf{w}_i (\mathbf{k}_i^\top \mathbf{k}_t) \right)$$

$$\mathbf{S}_t = \sum_{i=1}^{t} \mathbf{u}_i \mathbf{k}_i^\top, \qquad \mathbf{u}_t = \beta_t \left(\mathbf{v}_t - \sum_{i=1}^{t-1} \mathbf{u}_i (\mathbf{k}_i^\top \mathbf{k}_t) \right)$$

Parallelizing DeltaNet (Yang et al. 2024)



Check out our paper or blogpost (https://sustcsonglin.github.io/blog/2024/deltanet-2/) for more details.

Parallelizing DeltaNet (Yang et al. 2024)

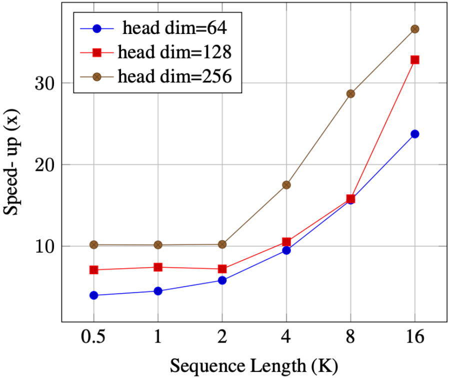


Figure: Speed-up of the chunkwise parallel form vs. the recurrent form.

When increasing the head dimension and sequence length, chunkwise implementation's speed-up is more significant.

Chunkwise training for Gated DeltaNet

$$\mathbf{S}_t = \mathbf{S}_{t-1}\left(\alpha_t\left(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top\right)\right) + \beta_t \mathbf{v}_t \mathbf{k}_t^\top$$
$$= \sum_{i=1}^{t} (\beta_i \mathbf{v}_i \mathbf{k}_i^\intercal \underbrace{\prod_{j=i+1}^{t} \alpha_j (\mathbf{I} - \beta_j \mathbf{k}_j \mathbf{k}_j^\intercal)}_{\text{defined as: } \mathbf{P}_i^t})$$

allows for *extended WY representation* where $\gamma_t := \prod_{i=1}^{t} \alpha_i$

$$\mathbf{P}_t = \gamma_t \left(\mathbf{I} - \sum_{i=1}^{t} \mathbf{w}_i \mathbf{k}_i^\intercal\right), \qquad \mathbf{w}_t = \beta_t \left(\mathbf{k}_t - \sum_{i=1}^{t-1} \mathbf{w}_i (\mathbf{k}_i^\intercal \mathbf{k}_t)\right)$$

$$\mathbf{S}_t = \sum_{i=1}^{t} \frac{\gamma_i}{\gamma_t} \mathbf{u}_i \mathbf{k}_i^\intercal, \qquad \mathbf{u}_t = \beta_t \left(\mathbf{v}_t - \sum_{i=1}^{t-1} \mathbf{u}_i \frac{\gamma_i}{\gamma_t} (\mathbf{k}_i^\intercal \mathbf{k}_t)\right)$$

The overheads of gating term is negligible and Gated DeltaNet is as fast as DeltaNet.

Chunkwise training for Gated DeltaNet

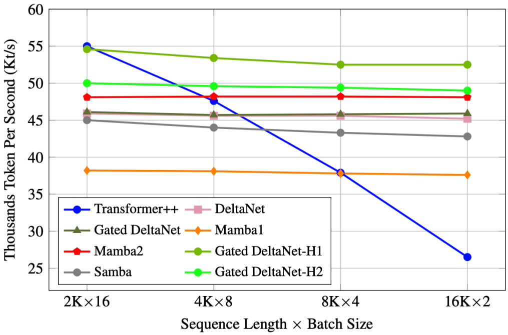


Figure: Training throughput of 1.3B models on a single H100.

- Gated DeltaNet is only slightly slower than Mamba2.
- Hybrid models have higher training throughput thanks to highly optimized flashattention kernel with sliding window size 2K.

Chunkwise training for Gated DeltaNet

This chunkwise algorithm can be further extended to the following linear recurrence with diagonal-plus-low-rank transition:

$$\mathbf{S}_t = \mathbf{S}_{t-1}(\mathbf{D}_t + \boldsymbol{\alpha}_t \boldsymbol{\beta}_t^\top) + \mathbf{v}_t \mathbf{k}_t^\top$$

- $\mathbf{D}_t \in \mathbb{R}^{d \times d}$ is a diagonal matrix. $\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t \in \mathbb{R}^d$ are vectors.
- RWKV-7 used such a linear recurrence and has been shown to be effective.
- Fast implementation is available in the flash-linear-attention library (https://github.com/fla-org/flash-linear-attention/blob/main/fla/ops/rwkv7/chunk.py).

Going beyond online linear regression objective

Going beyond online linear regression objective

Recall that DeltaNet optimizes the online linear regression loss:

$$\mathcal{L}_t(\mathbf{S}) = \frac{1}{2}\|\mathbf{S}\mathbf{k}_t - \mathbf{v}_t\|^2$$

- This optimization objective assumes linear relationships in historical data dependencies
- However, generative AI tasks involve complex, nonlinear dependencies
- A linear regression loss may be insufficient to capture these rich patterns.

Going beyond online linear regression objective

TTT (Sun et al. 2024a) extends this to a nonlinear regression loss:

$$\mathcal{L}_t(\mathbf{S}) = \frac{1}{2}\|f_{\mathbf{S}}(\mathbf{k}_t) - \mathbf{v}_t\|^2$$

where $f_{\mathbf{S}}$ is a nonlinear transformation parameterized by $\mathbf{S}$.

- TTT-linear: $f_{\mathbf{S}}(x) = \mathrm{LN}(\mathbf{S}x) + x$ where LN is layer normalization
- TTT-MLP: $f_{\mathbf{S}}(x) = \mathrm{LN}(\mathrm{MLP}_{\mathbf{S}}(x)) + x$ where $\mathbf{S}$ is MLP weight matrix

Going beyond online linear regression objective

TTT (Sun et al. 2024a) extends this to a nonlinear regression loss:

$$\mathcal{L}_t(\mathbf{S}) = \frac{1}{2}\|f_{\mathbf{S}}(\mathbf{k}_t) - \mathbf{v}_t\|^2$$

where $f_{\mathbf{S}}$ is a nonlinear transformation parameterized by $\mathbf{S}$.

- The nonlinear transformations increase expressivity but break the linear recurrence structure.
- Workaround: Use mini-batch updates by accumulating gradients over B tokens before updating $\mathbf{S}$ (i.e., hybrid intra-chunk linear + inter-chunk nonlinear).

Going beyond online linear regression objective

TTT (Sun et al. 2024a) extends this to a nonlinear regression loss:

$$\mathcal{L}_t(\mathbf{S}) = \frac{1}{2}\|f_{\mathbf{S}}(\mathbf{k}_t) - \mathbf{v}_t\|^2$$

where $f_{\mathbf{S}}$ is a nonlinear transformation parameterized by $\mathbf{S}$.

- Titans (Behrouz, Zhong, and Mirrokni 2024) further improves TTT by incorporating momentum and weight decay into the mini-batch SGD update.

Summary

- Modern RNNs through the lens of online learning:
  - (Decaying) Linear attention (RetNet, Lightning Attention, Mamba2, GLA, $\cdots$): negative inner-product loss
  - (Gated) DeltaNet: linear regression loss
  - TTT & Titans: nonlinear regression losses
- Gradient-based optimization techniques prove valuable:
  - Weight decay enables effective forgetting (Mamba2, Gated DeltaNet, $\cdots$)
  - Momentum improves performance (Titans)
- Efficient hardware utilization via:
  - Chunkwise training for linear attention.
  - Hybrid linear/nonlinear approaches across chunks (TTT & Titans)
- Promising future in bridging in-context meta learning and RNN architectures

Thanks!

References I

Arora, Simran et al. (2023). "Zoology: Measuring and Improving Recall in Efficient Language Models". In: *CoRR* abs/2312.04927.

Bai, Yushi et al. (2023). "LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding". In: *ArXiv* abs/2308.14508. URL: https://api.semanticscholar.org/CorpusID:261245264.

Beck, Maximilian et al. (2024). "xLSTM: Extended Long Short-Term Memory". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. URL: https://openreview.net/forum?id=ARAxPPIAhq.

Behrouz, Ali, Peilin Zhong, and Vahab Mirrokni (2024). *Titans: Learning to Memorize at Test Time*. arXiv: 2501.00663 [cs.LG]. URL: https://arxiv.org/abs/2501.00663.

References II

Bischof, Christian H. and Charles Van Loan (1985). "The WY representation for products of householder matrices". In: *SIAM Conference on Parallel Processing for Scientific Computing*. URL: https://api.semanticscholar.org/CorpusID:36094006.

Chou, Yuhong et al. (2024). "MetaLA: Unified Optimal Linear Approximation to Softmax Attention Map". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. URL: https://openreview.net/forum?id=Y8YVCOMEpz.

Dao, Tri and Albert Gu (2024). "Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality". In: *Forty-first International Conference on Machine Learning*. URL: https://openreview.net/forum?id=ztn8FCR1td.

Grazzi, Riccardo et al. (2024). "Unlocking State-Tracking in Linear RNNs Through Negative Eigenvalues". In: URL: https://api.semanticscholar.org/CorpusID:274141450.

References III

Gu, Albert and Tri Dao (2023). "Mamba: Linear-Time Sequence Modeling with Selective State Spaces". In.

Hsieh, Cheng-Ping et al. (2024). "RULER: What's the Real Context Size of Your Long-Context Language Models?" In: *ArXiv* abs/2404.06654. URL: https://api.semanticscholar.org/CorpusID:269032933.

Katharopoulos, Angelos et al. (2020). "Transformers are rnns: Fast autoregressive transformers with linear attention". In: *International conference on machine learning*. PMLR, pp. 5156–5165.

Liu, Bo et al. (2024). "Longhorn: State Space Models are Amortized Online Learners". In: *ArXiv* abs/2407.14207. URL: https://api.semanticscholar.org/CorpusID:271310065.

References IV

- Mao, Huanru Henry (Dec. 2022). "Fine-Tuning Pre-trained Transformers into Decaying Fast Weights". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, pp. 10236–10242. DOI: 10.18653/v1/2022.emnlp-main.697.
- Mereghetti, Carlo and Beatrice Palano (2000). "Threshold circuits for iterated matrix product and powering". In: *RAIRO Theor. Informatics Appl.* 34, pp. 39–46. URL: https://api.semanticscholar.org/CorpusID:13237763.
- Merrill, William, Jackson Petty, and Ashish Sabharwal (2024). "The Illusion of State in State-Space Models". In: *ArXiv* abs/2404.08819. URL: https://api.semanticscholar.org/CorpusID:269149086.
- MiniMax et al. (2025). *MiniMax-01: Scaling Foundation Models with Lightning Attention*. arXiv: 2501.08313 [cs.CL]. URL: https://arxiv.org/abs/2501.08313.

References V

- Peng, Bo et al. (2024). "Eagle and Finch: RWKV with Matrix-Valued States and Dynamic Recurrence". In.
- Poli, Michael et al. (2024). *Mechanistic Design and Scaling of Hybrid Architectures*. URL: https://arxiv.org/abs/2403.17844.
- Qin, Zhen et al. (2022). "The devil in linear transformer". In: *arXiv preprint arXiv:2210.10340*.
- Qin, Zhen et al. (2024a). "HGRN2: Gated Linear RNNs with State Expansion". In: URL: https://api.semanticscholar.org/CorpusID:269043328.
- Qin, Zhen et al. (2024b). "Various Lengths, Constant Speed: Efficient Language Modeling with Lightning Attention". In: *ArXiv* abs/2405.17381. URL: https://api.semanticscholar.org/CorpusID:270063820.
- Schlag, Imanol, Kazuki Irie, and Jürgen Schmidhuber (2021). "Linear Transformers Are Secretly Fast Weight Programmers". In: *International Conference on Machine Learning*. URL: https://api.semanticscholar.org/CorpusID:235377069.

References VI

- Sun, Yu et al. (2024a). "Learning to (Learn at Test Time): RNNs with Expressive Hidden States". In: *ArXiv* abs/2407.04620. URL: https://api.semanticscholar.org/CorpusID:271039606.
- Sun, Yutao et al. (2023). "Retentive network: A successor to transformer for large language models". In: *arXiv preprint arXiv:2307.08621*.
- Sun, Yutao et al. (2024b). "You Only Cache Once: Decoder-Decoder Architectures for Language Models". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. URL: https://openreview.net/forum?id=25Ioxw576r.
- Yang, Songlin, Jan Kautz, and Ali Hatamizadeh (2024). *Gated Delta Networks: Improving Mamba2 with Delta Rule*. arXiv: 2412.06464 [cs.CL]. URL: https://arxiv.org/abs/2412.06464.

References VII

Yang, Songlin et al. (2023). "Gated Linear Attention Transformers with Hardware-Efficient Training". In: *CoRR* abs/2312.06635. DOI: 10.48550/ARXIV.2312.06635. arXiv: 2312.06635. URL: https://doi.org/10.48550/arXiv.2312.06635.

Yang, Songlin et al. (2024). "Parallelizing Linear Transformers with the Delta Rule over Sequence Length". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. URL: https://openreview.net/forum?id=y8Rm4VNRPH.

Zhang, Yu et al. (2024). "Gated Slot Attention for Efficient Linear-Time Sequence Modeling". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. URL: https://openreview.net/forum?id=jY4PhQibmg.