Gated DeltaNet-2: Decoupling Erase and Write in Linear Attention

Ali Hatamizadeh **Yejin Choi** **Jan Kautz**
NVIDIA
{ahatamizadeh,yejinc,jkautz}@nvidia.com

Abstract

Linear attention replaces the unbounded cache of softmax attention with a fixed-size recurrent state, reducing sequence mixing to linear time and decoding to constant memory. The hard part is not just what to forget, but how to edit this compressed memory without scrambling existing associations. Delta-rule models subtract the current read before writing a new value, and Kimi Delta Attention (KDA) sharpens forgetting with channel-wise decay. But the active edit still uses a single scalar gate to control two different things, how much old content to erase on the key side and how much new content to commit on the value side. We introduce Gated DeltaNet-2, which generalizes both Gated DeltaNet and KDA by inheriting adaptive forgetting and channel-wise decay while addressing their shared limitation, the scalar tie between erasing and writing. Gated Delta Rule-2 separates these roles with a channel-wise erase gate $\boldsymbol{b}_t$ and a channel-wise write gate $\mathbf{w}_t$, reducing to KDA when both gates collapse to the same scalar and to Gated DeltaNet when the decay also collapses. We derive a fast-weight update view, a chunkwise WY algorithm with channel-wise decay absorbed into asymmetric erase factors, and a gate-aware backward pass that preserves efficient parallel training. At 1.3B parameters trained on 100B FineWeb-Edu tokens, Gated DeltaNet-2 achieves the strongest overall results among Mamba-2, Gated DeltaNet, KDA, and Mamba-3 variants across language modeling, commonsense reasoning, and retrieval. Its advantage is most pronounced on long-context RULER needle-in-a-haystack benchmarks, where it improves the evaluated multi-key retrieval setting and remains strong in both recurrent and hybrid settings.
Code: https://github.com/NVlabs/GatedDeltaNet-2

1 Introduction

The Transformer architecture has become the dominant backbone for large language models because self-attention gives each token direct access to its history and maps naturally to parallel training on modern accelerators. Its cost, however, still grows quadratically with sequence length. This cost becomes a central obstacle for long-context training and high-throughput inference, where the model must repeatedly process histories that are much longer than the dimension of a single attention head.

Linear recurrent attention takes a different path. It replaces the explicit attention matrix with a fixed-size recurrent state, turning sequence mixing into a linear-time recurrence whose memory does not grow with context length [1]. The appeal is clear, but so is the constraint. The state is a compressed key-value memory, and long contexts force many associations to share the same finite space, making exact retrieval difficult [2–7]. Recent work has improved this memory by giving the recurrence more control over what persists. Mamba-2 uses data-dependent decay to regulate the memory horizon [8]. DeltaNet replaces additive writes with the delta rule, enabling targeted overwrite of the association addressed by the current key [9, 2, 10]. Gated DeltaNet combines the delta rule with a learned decay gate, giving the state both global forgetting and targeted editing [11]. Kimi Delta

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Attention (KDA) refines the decay side with channel-wise forgetting over the key dimension [12]. In parallel, Mamba-3 advances the state-space route through exponential-trapezoidal discretization, complex-valued state transitions, and a multi-input, multi-output formulation for stronger and more efficient recurrence [13]. These advances have pushed recurrent linear models forward, while making the remaining bottleneck in delta-rule memory more visible. The active edit still uses one scalar gate to control both erasing old content and writing new content.

We propose *Gated DeltaNet-2*, a recurrent attention layer that decouples erase and write in the delta rule. The scalar tie is a modeling restriction because erasing and writing act on different axes of the state. Erasing is a key-side operation that decides which coordinates of the old read should be removed, while writing is a value-side operation that decides which coordinates of the incoming value should be committed. Gated DeltaNet-2 preserves KDA's channel-wise decay, but replaces the tied scalar delta gate with a channel-wise erase gate on the key axis and a channel-wise write gate on the value axis. The model can clear broad context through decay, remove selected stale associations through erase, and insert only the value channels that should persist through write. When the erase and write gates are tied to the same scalar, Gated DeltaNet-2 recovers KDA. If the decay is tied to a scalar as well, it recovers Gated DeltaNet.

This change preserves the efficient training path. By absorbing cumulative channel-wise decay into the rank-one erase factors, the recurrence admits a compact WY form with the same high-level chunkwise structure used by efficient delta-rule kernels [14–17]. The main text gives the modeling equations and the chunkwise algorithm. Kernel-level details are deferred to the supplement.

Empirically, Gated DeltaNet-2 improves the recurrent attention frontier, with the clearest gains on long-context retrieval. On the RULER needle-in-a-haystack tasks in Table 3, it remains strong as context length grows and is especially effective on the evaluated multi-key case where a fixed-size state must separate competing associations. This advantage also appears in real-world recall, where Gated DeltaNet-2 gives the strongest overall retrieval profile in both recurrent and hybrid settings. Together with gains in language modeling, commonsense reasoning and in-context retrieval, these results suggest that decoupling the active memory edit directly targets the main pressure point of fixed-state recurrence, interference among many compressed associations.

2 Preliminary

2.1 Linear attention as a recurrent state

We work with one attention head and omit layer indices. Let $\boldsymbol{q}_t, \boldsymbol{k}_t \in \mathbb{R}^{d_k}$ and $\boldsymbol{v}_t \in \mathbb{R}^{d_v}$ denote the query, key, and value at position t. A recurrent linear attention layer stores a matrix state $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$ and reads it with the query,

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top, \qquad \boldsymbol{o}_t = \mathbf{S}_t^\top \boldsymbol{q}_t. \tag{1}$$

This is the recurrent form of linear attention [1]. Expanding the recurrence over a length L sequence gives the familiar causal matrix form

$$\mathbf{O} = (\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V}, \tag{2}$$

where $\mathbf{M}$ is the causal mask. The state has fixed size in L, and the parallel form replaces tokenwise recurrence with matrix multiplication. The limitation is equally direct. Every outer product is added to the state and none is removed, so old associations remain until they are overwritten indirectly by later superposition.

Chunkwise form Efficient linear recurrent layers use a chunkwise schedule during training [15–17]. Split the sequence into chunks of size C. For chunk n, let $\mathbf{Q}_{[n]}, \mathbf{K}_{[n]}, \mathbf{V}_{[n]}$ be the query, key, and value blocks, and let $\mathbf{S}_{[n]}$ be the state at the start of the chunk. Partial expansion gives

$$\mathbf{S}_{[n+1]} = \mathbf{S}_{[n]} + \mathbf{K}_{[n]}^\top \mathbf{V}_{[n]}, \qquad \mathbf{O}_{[n]} = \mathbf{Q}_{[n]}\mathbf{S}_{[n]} + (\mathbf{Q}_{[n]}\mathbf{K}_{[n]}^\top \odot \mathbf{M}_C)\mathbf{V}_{[n]}. \tag{3}$$

The recurrence remains only across chunks, while all token interactions inside a chunk are expressed as dense matrix products. With a fixed C, this keeps linear complexity in sequence length and maps well to tensor cores.

2.2 Forgetting and overwriting

Mamba-2 adds a data-dependent scalar decay before each write [8],

$$\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top, \qquad \alpha_t \in (0, 1]. \tag{4}$$

The decay gives the model a global forgetting operation. If $\gamma_t = \prod_{i=1}^{t} \alpha_i$, then each earlier write is read at time t with factor γ_t/γ_i. This yields a decay-aware attention mask and preserves the chunkwise structure of Eq. 3.

DeltaNet instead gives the state an active edit operation [9, 2, 10]. Before writing $\boldsymbol{v}_t$, the model reads the value currently associated with $\boldsymbol{k}_t$ and subtracts it from the state. With a scalar step size $\beta_t \in [0, 1]$, the update is

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t (\boldsymbol{v}_t - \mathbf{S}_{t-1}^\top \boldsymbol{k}_t)^\top = (\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top)\mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top. \tag{5}$$

When $\|\boldsymbol{k}_t\|_2 = 1$, the matrix $\boldsymbol{k}_t \boldsymbol{k}_t^\top$ is a projector, so $\beta_t = 1$ overwrites the association at key $\boldsymbol{k}_t$ and $\beta_t = 0$ leaves it unchanged. In the fast-weight view [18, 19], Eq. 5 is one online gradient step on the local regression loss $\frac{1}{2}\|\mathbf{S}^\top \boldsymbol{k}_t - \boldsymbol{v}_t\|_2^2$.

Gated DeltaNet combines these two operations [11],

$$\mathbf{S}_t = \alpha_t (\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top)\mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top. \tag{6}$$

The decay clears the state uniformly, while the delta rule edits a selected association. This is a useful division of labor, but both gates are scalar per head.

KDA refines the decay side by replacing the scalar α_t with a channel-wise vector $\boldsymbol{\alpha}_t \in (0, 1]^{d_k}$ [12]. With $\mathbf{D}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t)$, its update can be written as

$$\mathbf{S}_t = (\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top)\mathbf{D}_t \mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top. \tag{7}$$

KDA lets each key channel decay at its own rate and retains the efficient WY-based chunkwise algorithm of DeltaNet [14, 10]. Yet the active gate β_t is still a single scalar. It controls both how much old content is erased from the read direction and how much new value is written. Gated DeltaNet-2 starts from this remaining tie.

3 Gated DeltaNet-2

3.1 Decoupling erase and write

KDA refines Gated DeltaNet by making the decay channel-wise, but the scalar β_t in Eq. 7 still carries two decisions that need not agree. One decision lives on the key side and determines which coordinates of the current read should be erased. The other lives on the value side and determines which coordinates of the candidate value should be written. Treating both decisions as one scalar is a restriction of the update, not a requirement of the delta rule.

Gated DeltaNet-2 separates the two decisions through *Gated Delta Rule-2*. Let

$$\boldsymbol{e}_t = \boldsymbol{b}_t \odot \boldsymbol{k}_t, \qquad \boldsymbol{z}_t = \mathbf{w}_t \odot \boldsymbol{v}_t, \tag{8}$$

where $\boldsymbol{b}_t \in [0, 1]^{d_k}$ is the erase gate and $\mathbf{w}_t \in [0, 1]^{d_v}$ is the write gate. The erase gate weights the key coordinates used to read old content, while the write gate weights the value coordinates being inserted. Let $\mathbf{D}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t)$. Applying decay before the active edit gives

$$\bar{\mathbf{S}}_t = \mathbf{D}_t \mathbf{S}_{t-1}, \qquad \boldsymbol{r}_t = \bar{\mathbf{S}}_t^\top \boldsymbol{e}_t, \qquad \mathbf{S}_t = \bar{\mathbf{S}}_t + \boldsymbol{k}_t (\boldsymbol{z}_t - \boldsymbol{r}_t)^\top. \tag{9}$$

Equivalently,

$$\boxed{\mathbf{S}_t = \big(\mathbf{I} - \boldsymbol{k}_t (\boldsymbol{b}_t \odot \boldsymbol{k}_t)^\top\big)\mathbf{D}_t \mathbf{S}_{t-1} + \boldsymbol{k}_t (\mathbf{w}_t \odot \boldsymbol{v}_t)^\top} \tag{10}$$

We refer to Eq. 10 as *Gated Delta Rule-2*. The output is $\boldsymbol{o}_t = \mathbf{S}_t^\top \boldsymbol{q}_t$. The left factor of the erase matrix remains $\boldsymbol{k}_t$, which preserves the write direction of the delta rule. The right factor becomes $\boldsymbol{b}_t \odot \boldsymbol{k}_t$, which makes the read direction channel selective. The write term becomes $\boldsymbol{k}_t \boldsymbol{z}_t^\top$, which makes the value update channel selective.

Gated Delta Rule-2 recovers KDA exactly when $\boldsymbol{b}_t = \beta_t \mathbf{1}_{d_k}$ and $\mathbf{w}_t = \beta_t \mathbf{1}_{d_v}$. It recovers Gated DeltaNet by further setting $\boldsymbol{\alpha}_t = \alpha_t \mathbf{1}_{d_k}$. Thus the model preserves the known scalar-gated updates as tied subspaces, while learning outside those subspaces when erase and write require different channel structure.

The layer produces the two gates with independent projections of the token representation,

$$\boldsymbol{b}_t = \sigma(\mathbf{W}_b \boldsymbol{x}_t), \qquad \mathbf{w}_t = \sigma(\mathbf{W}_w \boldsymbol{x}_t). \tag{11}$$

The log-decay follows the Gated DeltaNet parameterization,

$$\boldsymbol{g}_t = -\exp(\mathbf{a}) \odot \mathrm{softplus}(\mathbf{W}_f \boldsymbol{x}_t + \boldsymbol{\delta}), \qquad \boldsymbol{\alpha}_t = \exp(\boldsymbol{g}_t). \tag{12}$$

In practice this decay activation is computed in fp32 before the kernel consumes it, which avoids precision loss in the cumulative log-decay. We also support the negative-eigenvalue variant of Grazzi et al. [20] by scaling only the erase gate to $[0, 2]^{d_k}$. The write gate remains in $[0, 1]^{d_v}$ because the spectral effect concerns the state transition, not the value magnitude.

3.2 Fast-weight update perspective

We can interpret Gated Delta Rule-2 as an online update of a fast-weight memory state [21]. The state $\mathbf{S}_t$ stores transient key-value associations. At each token, the model first forms a decayed state $\bar{\mathbf{S}}_t = \mathbf{D}_t \mathbf{S}_{t-1}$, reads the old content through the gated erase direction $\boldsymbol{e}_t$, and writes a correction toward the gated value target $\boldsymbol{z}_t$.

More formally, Eq. 9 is the solution of the local online problem

$$\mathbf{S}_t = \arg\min_{\mathbf{S}} \boldsymbol{L}_t(\mathbf{S}), \qquad \boldsymbol{L}_t(\mathbf{S}) = \|\mathbf{S} - \bar{\mathbf{S}}_t\|_F^2 - 2\left\langle \mathbf{S}^\top \boldsymbol{k}_t, \boldsymbol{z}_t - \bar{\mathbf{S}}_t^\top \boldsymbol{e}_t \right\rangle. \tag{13}$$

The first term keeps the new state close to the decayed memory. The second term applies an associative edit whose residual compares the gated write target $\boldsymbol{z}_t$ against the content read from $\bar{\mathbf{S}}_t$ along $\boldsymbol{e}_t$. Since

$$\nabla_{\mathbf{S}} \boldsymbol{L}_t(\mathbf{S}) = 2(\mathbf{S} - \bar{\mathbf{S}}_t) - 2\boldsymbol{k}_t \left(\boldsymbol{z}_t - \bar{\mathbf{S}}_t^\top \boldsymbol{e}_t\right)^\top, \tag{14}$$

the minimizer is

$$\mathbf{S}_t = \bar{\mathbf{S}}_t + \boldsymbol{k}_t \left(\boldsymbol{z}_t - \bar{\mathbf{S}}_t^\top \boldsymbol{e}_t\right)^\top, \tag{15}$$

which is exactly Eq. 9.

Table 1 compares this view with Mamba-2, Gated DeltaNet, KDA, and Mamba-3. We write all updates in the state orientation used in this paper, where $\boldsymbol{o}_t = \mathbf{S}_t^\top \boldsymbol{q}_t$. Normalizer terms, kernel maps, output gates, and value projection gates are omitted for readability.

For the Mamba-3 row, we use the SISO exponential-trapezoidal recurrence [13]. Let

$$\widetilde{\boldsymbol{k}}_s = \mathbf{R}_{1:s}^\top \boldsymbol{k}_s, \qquad \eta_t = (1 - \lambda_t)\Delta_t \alpha_t, \qquad \zeta_t = \lambda_t \Delta_t. \tag{16}$$

Here $\mathbf{R}_{1:s}$ is the cumulative data-dependent rotation from the complex SSM view, and the previous-token term is omitted at the beginning of a sequence. The MIMO version replaces each rank-one write with a sum over the MIMO rank and leaves the same online form intact.

The comparison separates two families. Mamba-2 and Mamba-3 write correlations into a decayed state. Mamba-3 makes this write more expressive through the exponential-trapezoidal input rule and data-dependent rotations, but it does not subtract a current read from the state. Gated DeltaNet and KDA instead perform a residual delta edit. KDA changes the decay from scalar to channel-wise while keeping the scalar residual $\beta_t(\boldsymbol{v}_t - \bar{\mathbf{S}}_t^\top \boldsymbol{k}_t)$. Gated DeltaNet-2 changes the residual itself to

$$\boldsymbol{z}_t - \bar{\mathbf{S}}_t^\top \boldsymbol{e}_t = \mathbf{w}_t \odot \boldsymbol{v}_t - (\mathbf{D}_t \mathbf{S}_{t-1})^\top (\boldsymbol{b}_t \odot \boldsymbol{k}_t), \tag{17}$$

which decouples the coordinates used to erase from the coordinates used to write.

Table 1: Fast-weight update view of DeltaNet, Mamba-2, Gated DeltaNet, KDA, Mamba-3, and Gated DeltaNet-2. All updates use the state orientation of this paper, where $\boldsymbol{o}_t = \mathbf{S}_t^\top \boldsymbol{q}_t$. Mamba-2 and Mamba-3 add gated key-value correlation terms to a decayed state. DeltaNet, Gated DeltaNet, KDA, and Gated DeltaNet-2 instead write a delta residual, the target value minus the value currently read from memory.

Method	**Local objective $\boldsymbol{L}_t(\mathbf{S})$**	**State update**
DeltaNet [10]	$\|\mathbf{S}-\mathbf{S}_{t-1}\|_F^2$ $-2\left\langle \mathbf{S}^\top \boldsymbol{k}_t, \beta_t\left(\boldsymbol{v}_t - \mathbf{S}_{t-1}^\top \boldsymbol{k}_t\right)\right\rangle$	$\mathbf{S}_t = (\mathbf{I}-\beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top)\mathbf{S}_{t-1}$ $+\beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$
Mamba-2 [8]	$\|\mathbf{S}-\alpha_t\mathbf{S}_{t-1}\|_F^2$ $-2\left\langle \mathbf{S}^\top \boldsymbol{k}_t, \boldsymbol{v}_t\right\rangle$	$\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top$
Gated DeltaNet [11]	$\|\mathbf{S}-\alpha_t\mathbf{S}_{t-1}\|_F^2$ $-2\left\langle \mathbf{S}^\top \boldsymbol{k}_t, \beta_t\left(\boldsymbol{v}_t - (\alpha_t\mathbf{S}_{t-1})^\top \boldsymbol{k}_t\right)\right\rangle$	$\mathbf{S}_t = \alpha_t(\mathbf{I}-\beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top)\mathbf{S}_{t-1}$ $+\beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$
KDA [12]	$\|\mathbf{S}-\mathbf{D}_t\mathbf{S}_{t-1}\|_F^2$ $-2\left\langle \mathbf{S}^\top \boldsymbol{k}_t, \beta_t\left(\boldsymbol{v}_t - (\mathbf{D}_t\mathbf{S}_{t-1})^\top \boldsymbol{k}_t\right)\right\rangle$	$\mathbf{S}_t = (\mathbf{I}-\beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top)\mathbf{D}_t\mathbf{S}_{t-1}$ $+\beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$
Mamba-3 [13]	$\|\mathbf{S}-\alpha_t\mathbf{S}_{t-1}\|_F^2$ $-2\left\langle \mathbf{S}^\top \widetilde{\boldsymbol{k}}_{t-1}, \eta_t \boldsymbol{v}_{t-1}\right\rangle$ $-2\left\langle \mathbf{S}^\top \widetilde{\boldsymbol{k}}_t, \zeta_t \boldsymbol{v}_t\right\rangle$	$\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \eta_t \widetilde{\boldsymbol{k}}_{t-1} \boldsymbol{v}_{t-1}^\top$ $+\zeta_t \widetilde{\boldsymbol{k}}_t \boldsymbol{v}_t^\top$
Gated DeltaNet-2	$\|\mathbf{S}-\mathbf{D}_t\mathbf{S}_{t-1}\|_F^2$ $-2\left\langle \mathbf{S}^\top \boldsymbol{k}_t, \boldsymbol{z}_t - (\mathbf{D}_t\mathbf{S}_{t-1})^\top \boldsymbol{e}_t\right\rangle$	$\mathbf{S}_t = (\mathbf{I}-\boldsymbol{k}_t \boldsymbol{e}_t^\top)\mathbf{D}_t\mathbf{S}_{t-1}$ $+\boldsymbol{k}_t \boldsymbol{z}_t^\top$

3.3 Chunkwise parallel training

We now show that Gated Delta Rule-2 keeps the same chunkwise structure as KDA. Consider one chunk and suppress the chunk index. Let

$$\boldsymbol{G}_r = \sum_{i=1}^{r} \boldsymbol{g}_i, \qquad \boldsymbol{\gamma}_r = \exp(\boldsymbol{G}_r), \qquad \boldsymbol{\gamma}_0 = \mathbf{1}_{d_k}. \tag{18}$$

Define the decay-normalized state $\widehat{\mathbf{S}}_r$ by $\mathbf{S}_r = \mathrm{Diag}(\boldsymbol{\gamma}_r)\widehat{\mathbf{S}}_r$. With $\widehat{\mathbf{S}}_0 = \mathbf{S}_{[n]}$, Eq. 10 becomes a pure asymmetric delta recurrence,

$$\widehat{\mathbf{S}}_r = \left(\mathbf{I} - \bar{\boldsymbol{k}}_r \bar{\boldsymbol{e}}_r^\top\right)\widehat{\mathbf{S}}_{r-1} + \bar{\boldsymbol{k}}_r \boldsymbol{z}_r^\top, \qquad \bar{\boldsymbol{k}}_r = \boldsymbol{\gamma}_r^{-1} \odot \boldsymbol{k}_r, \qquad \bar{\boldsymbol{e}}_r = \boldsymbol{\gamma}_r \odot \boldsymbol{e}_r. \tag{19}$$

This normalization is the key to the efficient form. The channel-wise decay is absorbed into the two factors of each rank-one erase, while the update remains a product of matrices of the form $\mathbf{I} - \bar{\boldsymbol{k}}_r \bar{\boldsymbol{e}}_r^\top$.

Let $\mathbf{B} \in \mathbb{R}^{C \times d_k}$ and $\mathbf{W} \in \mathbb{R}^{C \times d_v}$ contain rows $\boldsymbol{b}_r^\top$ and $\mathbf{w}_r^\top$, respectively. For compact matrix notation, let $\boldsymbol{\gamma} \in \mathbb{R}^{C \times d_k}$ contain rows $\boldsymbol{\gamma}_r^\top$. Let $\bar{\mathbf{K}} \in \mathbb{R}^{C \times d_k}$, $\bar{\mathbf{E}} \in \mathbb{R}^{C \times d_k}$, and $\mathbf{Z} \in \mathbb{R}^{C \times d_v}$ contain rows $\bar{\boldsymbol{k}}_r^\top$, $\bar{\boldsymbol{e}}_r^\top$, and $\boldsymbol{z}_r^\top$. Equivalently,

$$\bar{\mathbf{E}} = \boldsymbol{\gamma} \odot (\mathbf{B} \odot \mathbf{K}), \qquad \mathbf{Z} = \mathbf{W} \odot \mathbf{V}. \tag{20}$$

Define the strictly lower triangular matrix

$$\mathbf{T} = \mathrm{tril}(\bar{\mathbf{E}}\bar{\mathbf{K}}^\top, -1), \qquad \mathbf{A} = (\mathbf{I} + \mathbf{T})^{-1}. \tag{21}$$

The WY auxiliaries are

$$\mathbf{Y} = \mathbf{A}\bar{\mathbf{E}}, \qquad \mathbf{U} = \mathbf{A}\mathbf{Z}. \tag{22}$$

Here $\mathbf{Y}$ is the erase-side auxiliary and $\mathbf{U}$ is the write-side auxiliary. Since $\mathbf{T}$ is triangular with zero diagonal, $\mathbf{A}$ is obtained by a small forward substitution inside each chunk.

The end-of-chunk state is then

$$\mathbf{S}_{[n+1]} = \mathrm{Diag}(\boldsymbol{\gamma}_C)\mathbf{S}_{[n]} + \mathbf{K}_{\mathrm{tail}}^\top(\mathbf{U} - \mathbf{Y}\mathbf{S}_{[n]}), \tag{23}$$

where row r of $\mathbf{K}_{\mathrm{tail}}$ is $(\boldsymbol{\gamma}_C / \boldsymbol{\gamma}_r) \odot \boldsymbol{k}_r$. The output block is

$$\mathbf{O}_{[n]} = \mathbf{Q}_\gamma \mathbf{S}_{[n]} + \mathbf{A}_{qk}(\mathbf{U} - \mathbf{Y}\mathbf{S}_{[n]}), \tag{24}$$

where row r of $\mathbf{Q}_\gamma$ is $\boldsymbol{\gamma}_r \odot \boldsymbol{q}_r$ and

$$(\mathbf{A}_{qk})_{rs} = \mathbf{1}_{r \geq s}\, \boldsymbol{q}_r^\top \operatorname{Diag}(\boldsymbol{\gamma}_r / \boldsymbol{\gamma}_s) \boldsymbol{k}_s. \tag{25}$$

Equations 23 and 24 have the same shape as the KDA chunk equations. The only difference is how $\mathbf{Y}$ and $\mathbf{U}$ are formed. The erase gate enters through row r of $\bar{\mathbf{E}}$ as $\boldsymbol{\gamma}_r \odot (\boldsymbol{b}_r \odot \boldsymbol{k}_r)$. The write gate enters through row r of $\mathbf{Z}$ as $\mathbf{w}_r \odot \boldsymbol{v}_r$. The rest of the computation is a triangular solve and dense matrix multiplication over fixed-size chunks. We use the UT transform [22] and implement these equations with fused Triton kernels [23]. Kernel schedules and precision choices are deferred to the supplement.

3.4 Gate-aware backward

The backward pass follows the same decomposition as the forward. Gradients first flow through the output equation and the inter-chunk state recurrence, both of which operate only on $\mathbf{A}_{qk}$, $\mathbf{K}_{\text{tail}}$, $\mathbf{Y}$, and $\mathbf{U}$. The only new accounting is the vector-Jacobian product through Eq. 22 and Eq. 21.

For scalar-gated delta rules, a factor β_r can be moved outside the dot products that accumulate the gradient of $\mathbf{A}$. That shortcut breaks for Gated Delta Rule-2. The write side contains a different diagonal gate over value channels, and the erase side contains a different diagonal gate over key channels. Therefore the gate factors must be present at the accumulation sites. Let $\mathbf{B}$ and $\mathbf{W}$ contain rows $\boldsymbol{b}_r^\top$ and $\mathbf{w}_r^\top$, respectively, and let $\boldsymbol{\gamma}$ denote the row-stacked cumulative-decay vectors. Then

$$\mathrm{d}\mathbf{A} \mathrel{+}= \mathrm{d}\mathbf{U}\, \mathbf{Z}^\top, \qquad \mathbf{Z} = \mathbf{W} \odot \mathbf{V}, \tag{26}$$

$$\mathrm{d}\mathbf{A} \mathrel{+}= \mathrm{d}\mathbf{Y}\, \bar{\mathbf{E}}^\top, \qquad \bar{\mathbf{E}} = \boldsymbol{\gamma} \odot (\mathbf{B} \odot \mathbf{K}). \tag{27}$$

The inverse itself has the standard triangular vector-Jacobian product

$$\mathrm{d}\mathbf{T} = -\operatorname{tril}\big(\mathbf{A}^\top \mathrm{d}\mathbf{A}\, \mathbf{A}^\top, -1\big). \tag{28}$$

From there, gradients to $\mathbf{B}$, $\mathbf{W}$, $\mathbf{K}$, $\mathbf{V}$, and the cumulative decay follow by ordinary elementwise products and reverse cumulative sums. This gate-aware accumulation is the main mathematical change required for training Gated Delta Rule-2. The remaining backward kernels retain the same matrix shapes as KDA and can reuse the same state and output vector-Jacobian product structure.

3.5 Block design and hybrid models

Gated DeltaNet-2 token mixer. Gated DeltaNet-2 is used as the recurrent token mixer in a standard Transformer-style block. Fig. 1 (right) shows its block design. For the Gated Delta Rule-2 in Eq. 10, $\{\boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t\}$ are produced by linear projection, short causal convolution, and SiLU, with L2 normalization applied to $\boldsymbol{q}_t$ and $\boldsymbol{k}_t$ for stability. Separate branches produce the channel-wise decay $\boldsymbol{\alpha}_t$, erase gate $\boldsymbol{b}_t$, and write gate $\mathbf{w}_t$. The recurrent output is RMS-normalized, multiplied by a separate SiLU output gate, and projected back to the model dimension. Throughout the paper, $\boldsymbol{g}$ denotes the log-decay tensor in Eq. 12, not the output gate. With grouped value heads, $\boldsymbol{q}$, $\boldsymbol{k}$, the log-decay tensor $\boldsymbol{g}$, and $\boldsymbol{b}$ are repeated across value-head groups, while $\boldsymbol{v}$ and $\mathbf{w}$ remain on the value-head axis.

Model families. We train both recurrent and hybrid models. The recurrent model stacks Gated DeltaNet-2 token mixers and MLPs under the standard residual block, isolating the fixed-state memory of Eq. 10. The hybrid model inserts Sliding-Window Attention (SWA) after the recurrent mixer, as shown in Fig. 1 (left). A repeated cell contains Gated DeltaNet-2, an MLP, SWA, and another MLP. Gated DeltaNet-2 compresses long histories into constant-size memory, while SWA handles exact local interactions such as short shifts, comparisons, and local retrieval. With a fixed window, the hybrid retains linear sequence scaling and a bounded attention cache, following the recurrent attention hybrid design pattern [24, 25].

4 Experiments

Setup We evaluate each recurrent family in two forms, a recurrent-only model and a hybrid model that pairs the same recurrent token mixer with sliding-window attention as described in Section 3.5.

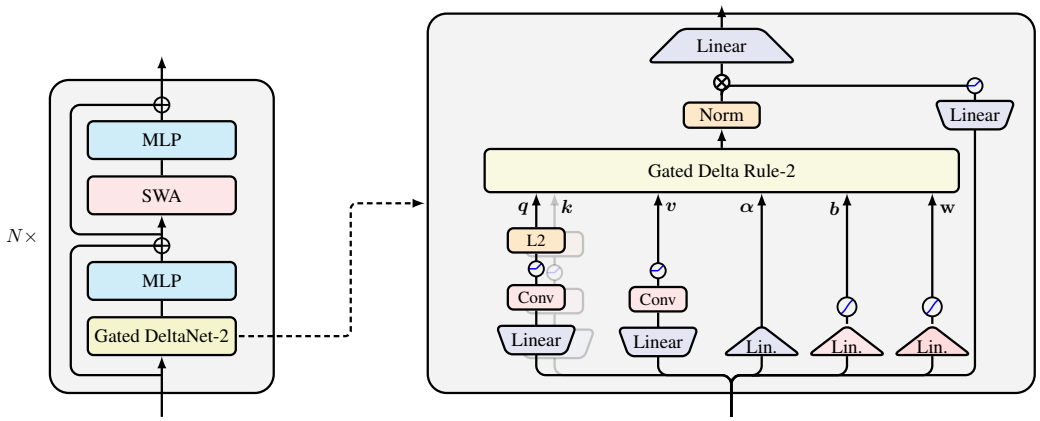


Figure 1: Visualization of the hybrid architecture and block design of **Gated DeltaNet-2**. The *Hybrid Gated DeltaNet-2* model repeats a Gated DeltaNet-2 token mixer, an MLP, sliding-window attention (SWA), and another MLP. In the block design, query and key paths use linear projection, short convolution, SiLU, and L2 normalization. The value path uses linear projection, short convolution, and SiLU. The central recurrent operator is *Gated Delta Rule-2*. The decay branch produces $\boldsymbol{\alpha}$ from the log-decay projection. The channel-wise erase gate $\boldsymbol{b}$ and channel-wise write gate $\mathbf{w}$ each use linear projection followed by sigmoid. The recurrent output is normalized, multiplied by a SiLU output gate, and passed through the output projection.

Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc ↑	OBQA acc ↑	SIQA acc ↑	BoolQ acc ↑	Avg. acc ↑
Recurrent models												
Mamba-2	16.79	12.38	45.24	72.58	55.51	55.33	70.68	35.26	31.00	40.63	60.19	51.82
Gated DeltaNet	16.40	11.89	**49.62**	72.31	56.50	56.75	68.81	35.15	30.20	40.53	58.78	52.07
KDA	16.81	11.68	48.13	72.09	55.75	55.72	70.83	35.92	30.40	40.99	**60.67**	52.28
Mamba-3 (SISO)	16.30	12.99	45.06	72.31	55.58	56.20	70.45	34.56	31.00	**41.76**	55.90	51.42
Mamba-3 (MIMO)	16.45	11.66	47.82	72.36	56.49	55.78	72.38	38.07	30.00	40.89	57.74	52.39
Gated DeltaNet-2	**15.90**	**11.41**	48.09	**72.80**	**56.84**	**57.85**	**72.43**	**38.23**	**31.60**	40.58	59.54	**53.11**
Attention or hybrid models												
Transformer	19.22	13.72	48.32	70.21	56.12	55.85	69.23	33.84	25.00	39.74	59.42	50.86
Mamba-2	17.46	11.29	48.05	71.47	57.52	56.17	70.50	34.73	29.80	40.35	59.31	51.99
Gated DeltaNet	16.00	10.82	48.71	70.06	57.50	56.83	70.41	35.15	30.60	40.97	60.00	52.25
KDA	16.01	10.66	49.21	71.06	56.89	57.77	71.59	35.07	30.00	40.53	62.03	52.68
Mamba-3 (SISO)	**15.54**	10.65	49.19	71.01	**58.75**	57.30	70.54	36.35	32.00	41.20	57.86	52.69
Mamba-3 (MIMO)	15.81	10.92	49.82	71.98	58.19	57.06	70.54	**38.48**	29.40	40.99	57.98	52.72
Gated DeltaNet-2	15.62	**10.43**	**50.90**	**72.20**	58.46	**58.56**	**71.89**	36.69	**33.00**	**41.50**	**62.57**	**53.97**

Table 2: Performance comparison on language modeling and zero-shot common-sense reasoning. All accuracy values are reported as percentages. Avg. is computed over LAMBADA accuracy and the listed reasoning accuracies.

For Mamba-3, we include both SISO and MIMO variants and use MIMO rank $R = 4$ following Lahoti et al. [13]. All models are trained with the same recipe. Unless stated otherwise, each model has 1.3B parameters and is trained on 100B tokens from FineWeb-Edu [26]. We use AdamW with peak learning rate 4×10^{-4}, weight decay 0.1, gradient clipping at 1.0, cosine decay, a 1B-token warm-up, and a global batch size of 0.5M tokens. The training length is 4K tokens, and hybrid models use a 2K sliding-window attention size. Evaluation details are given in the appendix.

Language modeling and common-sense reasoning Table 2 reports WikiText and LAMBADA perplexity [27, 28], zero-shot LAMBADA accuracy, and the common-sense suite from PIQA through BoolQ [29–35]. Gated DeltaNet-2 achieves the best average in both recurrent and hybrid settings. Since recurrent state size is matched, the gain points to a stronger update rule rather than a larger memory. The trend persists with SWA, and the model is more balanced than Mamba-3 across perplexity, accuracy, and transfer.

Model	S-NIAH-1				S-NIAH-2				S-NIAH-3			MK-NIAH-1		
	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	1K	2K	4K
Recurrent models														
Mamba-2	**100.0**	**100.0**	97.0	55.8	99.6	99.6	62.6	21.0	59.2	38.6	14.4	29.0	21.2	21.4
Gated DeltaNet	99.8	**100.0**	**100.0**	97.6	**100.0**	**100.0**	87.2	32.0	89.8	54.2	**60.6**	58.0	37.0	27.8
KDA	**100.0**	**100.0**	99.2	70.6	**100.0**	**100.0**	89.0	30.6	77.4	63.2	26.2	54.0	44.2	28.0
Mamba-3 (SISO)	**100.0**	99.0	63.4	27.8	99.8	99.0	59.4	25.2	60.2	35.6	12.2	44.8	27.4	20.2
Mamba-3 (MIMO)	**100.0**	99.8	93.0	35.6	99.8	98.8	64.2	27.2	89.2	72.4	29.2	49.4	19.2	18.0
Gated DeltaNet-2	**100.0**	**100.0**	**100.0**	**97.8**	**100.0**	**100.0**	**93.0**	**39.2**	**92.0**	**89.8**	31.8	**72.6**	**51.4**	**37.8**
Attention or hybrid models														
Transformer	**100.0**	**100.0**	51.2	0.0	**100.0**	**100.0**	44.2	0.0	95.8	94.8	37.0	75.6	66.6	38.2
Mamba-2	**100.0**	**100.0**	51.8	25.4	**100.0**	99.6	52.4	25.8	97.8	86.8	48.0	82.0	58.6	39.0
Gated DeltaNet	**100.0**	**100.0**	47.2	22.4	**100.0**	99.8	57.3	25.6	94.8	91.2	47.2	91.0	78.4	44.8
KDA	**100.0**	**100.0**	51.8	26.2	**100.0**	**100.0**	56.0	23.0	97.2	93.4	51.6	91.4	84.0	40.4
Mamba-3 (SISO)	**100.0**	**100.0**	49.6	26.0	**100.0**	**100.0**	**58.2**	27.8	95.0	90.4	44.0	78.8	65.6	33.6
Mamba-3 (MIMO)	**100.0**	**100.0**	49.0	22.8	**100.0**	**100.0**	53.0	27.8	99.4	98.4	54.2	82.4	79.0	46.6
Gated DeltaNet-2	**100.0**	**100.0**	**55.2**	**27.4**	**100.0**	**100.0**	57.9	**29.2**	**99.6**	**99.0**	**55.6**	**93.0**	**84.6**	**48.0**

Table 3: Accuracy on Single Needle-In-A-Haystack (S-NIAH) and Multi-Key Needle-In-A-Haystack (MK-NIAH) tasks from RULER. Best values within each model family and context length are bolded; second-best values are underlined.

In-context retrieval on synthetic data Table 3 reports S-NIAH and MK-NIAH from RULER [36], which test retention, interference control, high-entropy value storage, and multi-key discrimination under fixed-state memory. Gated DeltaNet-2 is strongest where memory editing matters most. In the recurrent setting, it leads the interference-heavy S-NIAH-2 cases at 4K and 8K and all MK-NIAH-1 lengths. The hybrid model shows the same pattern, leading the long S-NIAH-1 cases, the 8K S-NIAH-2 case, all S-NIAH-3 lengths, and the longer MK-NIAH-1 settings. These gains match the design of Gated Delta Rule-2. The key-side erase gate $\boldsymbol{b}_t$ selectively protects or revises key channels, while the value-side write gate $\mathbf{w}_t$ controls which value channels enter the state. With SWA handling local evidence, this decoupled recurrent update preserves longer-range associations more effectively than a scalar delta gate.

In-context retrieval on real-world tasks Table 4 reports recall-heavy real-world tasks from Arora et al. [37], spanning extraction, question answering, and distractor-rich evidence. These tasks are less controlled than synthetic NIAH but better reflect fixed-state memory under realistic context. Gated DeltaNet-2 achieves the best average in both recurrent and hybrid settings. Its recurrent gains are strongest on noisy association recovery, where selective erase and gated write are directly useful. The remaining NQ and DROP gaps point to formats that also need local evidence aggregation, which SWA supplies in the hybrid model.

Models	SWDE	SQD	FDA	TQA	NQ	DROP	Avg.
Recurrent models							
Mamba-2	17.24	32.38	14.53	58.35	18.91	19.60	26.84
Gated DeltaNet	17.90	32.67	18.52	59.60	**20.16**	19.69	28.09
Mamba-3 (SISO)	17.62	35.07	11.08	58.89	18.18	21.32	27.03
KDA	22.49	35.10	14.90	58.12	19.58	**21.80**	28.67
Mamba-3 (MIMO)	16.68	36.65	17.44	59.06	19.16	21.08	28.35
Gated DeltaNet-2	**23.65**	**36.75**	**19.98**	**61.37**	19.64	17.87	**29.88**
Attention or hybrid models							
Transformer	32.21	38.67	54.78	58.09	22.49	22.18	38.07
Mamba-2	34.67	40.74	52.31	60.13	25.91	**24.68**	39.74
Gated DeltaNet	33.18	42.28	50.86	60.60	25.78	21.95	39.11
Mamba-3 (SISO)	35.30	**46.42**	54.95	59.54	25.91	23.96	41.01
KDA	39.83	40.10	53.59	59.89	25.27	22.18	40.14
Mamba-3 (MIMO)	32.33	44.70	**55.31**	59.00	26.26	23.08	40.11
Gated DeltaNet-2	**41.96**	44.70	54.68	**62.38**	**26.31**	23.67	**42.28**

Table 4: Accuracy on real-world retrieval tasks with input length truncated to 2K tokens. SQD denotes SQuAD. TQA denotes TriviaQA.

Scaling to a global-attention hybrid The experiments above fix the scale at 1.3B parameters and 100B tokens and pair each recurrent mixer with sliding-window attention. This leaves open whether the benefit of decoupled erase and write survives a larger model and training budget, and whether the finer memory edit still matters once a model contains full global-attention layers, which give direct access to the entire context and thus remove part of the pressure on the recurrent state. To answer both questions, we train Nemotron-3 Nano-style hybrid latent-MoE models [38, 39] with 31.6B total and 3.2B active parameters on 300B tokens. Each model interleaves recurrent mixers and MoE layers with a small number of full global-attention layers. We keep the global-attention layers and their placement, the recurrent-state size, the active parameter count, and the training recipe fixed, and vary

Recurrent mixer	MMLU		Code			Math			Commonsense understanding					Overall
	Pro	Str.	HumanEval	MBPP	Avg.	GSM8K	MATH-500	Avg.	RACE	ARC-c	Hella.	Wino.	Avg.	Avg.
Mamba-2	37.41	61.27	42.71	56.32	49.52	71.87	60.80	66.34	75.50	77.56	77.78	70.56	75.35	60.14
KDA	37.16	60.30	42.83	58.72	50.78	74.68	64.40	69.54	**78.47**	78.16	77.94	69.14	75.93	61.25
Gated DeltaNet-2	**38.00**	**61.73**	**43.31**	**60.06**	**51.69**	**79.30**	**65.20**	**72.25**	78.18	**80.12**	**78.03**	**71.11**	**76.86**	**62.67**

Table 5: Accuracy of 31.6B-parameter hybrid latent-MoE models with 3.2B active parameters, trained on 300B tokens with full global-attention layers. Only the recurrent mixer is varied; the attention layers, recurrent-state size, active parameter count, and training recipe are fixed. Code Avg. averages HumanEval and MBPP, Math Avg. averages GSM8K and MATH-500, and the commonsense Avg. averages RACE, ARC-c, HellaSwag, and WinoGrande. Overall Avg. equally averages the MMLU, code, math, and commonsense category scores, where the MMLU score is the mean of its two columns. Best values are bolded.

Variant	Wiki. ppl ↓	LMB. ppl ↓	Common. avg ↑	S-NIAH-2 @4K ↑	S-NIAH-3 @2K ↑	MK-NIAH-1 @4K ↑	Recall avg ↑
Channel structure							
w-only, scalar $\boldsymbol{b}_t$, channel $\mathbf{w}_t$	16.55	11.62	52.45	90.6	71.4	30.6	28.92
b-only, channel $\boldsymbol{b}_t$, scalar $\mathbf{w}_t$	16.12	11.50	52.79	92.1	84.6	35.2	29.51
Erase range							
Gated DeltaNet-2, $\boldsymbol{b}_t \in [0,1]^{d_k}$	**15.90**	**11.41**	**53.11**	93.0	**89.8**	**37.8**	**29.88**
expanded $\boldsymbol{b}_t \in [0,2]^{d_k}$	15.95	11.44	53.04	**93.1**	89.4	37.6	29.81
Delta correction							
Read/write gates, no delta correction	16.28	11.60	52.58	85.4	72.1	30.9	29.08

Table 6: Gate structure, erase range, and delta-correction ablations in the recurrent-only setting. The no-delta variant keeps both channel-wise gate projections but replaces the read-before-write correction with an additive memory, Eq. 29.

only the recurrent mixer among Mamba-2, KDA, and Gated DeltaNet-2. Small adjustments to the FFN intermediate size keep the three variants iso-parameter. Table 5 reports accuracy on knowledge, code, math, and commonsense benchmarks, which are described in Appendix E.3.

Gated DeltaNet-2 is best on nine of the ten benchmarks and on every category average. It improves the overall average by 1.42 points over KDA and by 2.53 points over Mamba-2. The gains are largest on math, where the category average improves by 2.71 and 5.91 points, respectively, and they extend to code and commonsense understanding. Two observations follow. First, the advantage of decoupling erase and write does not disappear with greater model capacity or a threefold larger token budget. The relevant quantity is not the parameter count alone but the number of competing associations that must share a fixed-size state, and larger models trained on more data can also learn more selective token-dependent erase and write policies. Second, global attention does not make the recurrent update redundant. Even when a few attention layers provide direct long-range access, replacing the recurrent mixer with Gated DeltaNet-2 yields consistent gains at matched state size, parameter budget, and training tokens. We therefore expect similar gains in larger global-attention hybrids, and potentially larger gains in recurrent-only or bounded-attention models at longer contexts, where interference within the fixed-size state is more severe. The latter remains a mechanistic expectation rather than a claim established by these experiments.

Gate structure and erase range ablations. Table 6 evaluates two aspects of the Gated Delta Rule-2 update, the channel structure of the erase and write gates, and the range of the erase gate. For the channel-structure ablations, we average either gate over its channel axis and broadcast the scalar back at runtime, while keeping the original projections unchanged. Thus the parameter count stays fixed and only channel-wise gate variation is removed. Both scalarized variants trail full Gated DeltaNet-2, showing that both gates use their channel degrees of freedom. The asymmetry is clear. Keeping channel structure only in $\boldsymbol{b}_t$ recovers most of the full model on language modeling and retrieval, whereas keeping it only in $\mathbf{w}_t$ recovers less. This matches Eq. 10, where $\boldsymbol{b}_t$ changes the key-side erase factor $\boldsymbol{k}_t(\boldsymbol{b}_t \odot \boldsymbol{k}_t)^\top$, while $\mathbf{w}_t$ reweights the written value. Finally, expanding the erase range from $[0,1]^{d_k}$ to $[0,2]^{d_k}$ gives no consistent gain at this scale.

Removing the delta-rule correction. The ablations above all retain the delta rule, so they do not show whether the retrieval gains come from the read-before-write correction or from channel-wise

gating alone. We therefore train a parameter-matched control that keeps both channel-wise gate projections but replaces the correction with an additive memory. Without an internal state read, $\boldsymbol{b}_t$ becomes a channel-wise read gate on the query and $\mathbf{w}_t$ remains a channel-wise write gate,

$$\bar{\mathbf{S}}_t = \mathbf{D}_t\mathbf{S}_{t-1}, \qquad \mathbf{S}_t = \bar{\mathbf{S}}_t + \boldsymbol{k}_t(\mathbf{w}_t \odot \boldsymbol{v}_t)^\top, \qquad \boldsymbol{o}_t = \mathbf{S}_t^\top(\boldsymbol{b}_t \odot \boldsymbol{q}_t). \tag{29}$$

This control has the same recurrent-state size, parameter count, gate projections, and training recipe as Gated DeltaNet-2. The last block of Table 6 shows that it remains competitive on language modeling, but degrades substantially when memory interference requires targeted replacement. Gated DeltaNet-2 improves S-NIAH-2, S-NIAH-3, and MK-NIAH-1 by 7.6, 17.7, and 6.9 points, respectively. This isolates the benefit of the delta correction from the additional vector gates: the retrieval gains of Gated DeltaNet-2 cannot be explained by channel-wise read and write gating alone.

Throughput comparison. Fig. 2 reports single H100 training throughput for the hybrid 1.3B models under a fixed token budget. Gated DeltaNet-2 preserves the near-flat scaling profile of recurrent mixers as sequence length grows, dropping only mildly from 38.0 to 36.1 Kt/s, while the Transformer degrades sharply. Relative to KDA, the small gap reflects the added channel-wise erase and write gates. Thus Gated DeltaNet-2 retains practical training efficiency while paying a modest constant cost for finer memory control.

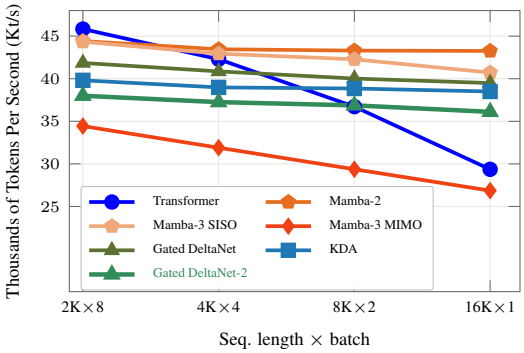


Figure 2: Training throughput on a H100 GPU.

5 Related Work

Efficient sequence models replace quadratic self-attention with recurrent or linear-time token mixers that maintain a fixed-size state. Early structured state-space and recurrent models used mostly data-independent transitions [40–42, 16], while Mamba and Mamba-2 introduced data-dependent selective dynamics and the SSD framework [43, 8]. Gated linear attention and related linear RNNs further improve memory control with learned decay gates [17, 44]. Delta-rule models take a complementary fast-weight view, where the recurrent state is updated by correcting the current read before writing the new value, improving associative memory over Hebbian-style accumulation [45–47, 10]. Gated DeltaNet adds adaptive forgetting to this update [11], and KDA strengthens it with channel-wise decay and an efficient chunkwise algorithm, but still uses a scalar β_t to control both erasing and writing [12]. Mamba-3 advances the SSM line instead, using exponential-trapezoidal discretization, complex-valued transitions implemented through data-dependent rotations, and a MIMO formulation for stronger modeling at efficient decoding latency [13]. Our work is complementary to these directions. Gated DeltaNet-2 keeps the delta-rule fast-weight structure of GDN and KDA, but replaces the tied scalar update strength with a channel-wise erase gate $\boldsymbol{b}_t$ and a channel-wise write gate $\mathbf{w}_t$. This recovers KDA when both gates are tied to the same scalar, while allowing old content and new values to be controlled along different channel patterns.

6 Conclusion

We introduced Gated DeltaNet-2, a delta-rule recurrent attention layer that decouples the active memory edit into channel-wise erase and write decisions. The erase gate $\boldsymbol{b}_t$ selects which key-side coordinates of the decayed state are read and removed, while the write gate $\mathbf{w}_t$ selects which value-side coordinates are committed. This removes the scalar β_t tie in Gated DeltaNet and KDA, recovers both as special cases, and preserves efficient chunkwise training through a WY form with gate-aware kernels. Under matched 1.3B training, Gated DeltaNet-2 improves the recurrent and hybrid frontier across language modeling, commonsense reasoning, synthetic retrieval, and real-world recall, while adding only a small constant throughput overhead. The gain persists in a 31.6B-parameter hybrid latent-MoE model with full global attention trained on 300B tokens, where Gated DeltaNet-2 leads KDA and Mamba-2 on every category average. Ablations show that both gates contribute, with the erase gate accounting for most of the gain.

Acknowledgements and Disclosure of Funding

This work was conducted at and funded by NVIDIA; all authors are employees of NVIDIA. No third-party funding supported this work in the last 36 months, and the authors declare no financial relationships outside the submitted work that could be perceived as influencing it.

References

[1] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR, 2020. URL http://proceedings.mlr.press/v119/katharopoulos20a.html.

[2] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 9355–9366. PMLR, 2021. URL http://proceedings.mlr.press/v139/schlag21a.html.

[3] Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LY3ukUANko.

[4] Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 1763–1840. PMLR, 2024. URL https://proceedings.mlr.press/v235/arora24a.html.

[5] Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 21502–21521. PMLR, 2024. URL https://proceedings.mlr.press/v235/jelassi24a.html.

[6] Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. Rnns are not transformers (yet): The key bottleneck on in-context retrieval. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=h3wbI8Uk1Z.

[7] Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Architectures and algorithms. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 787–812. PMLR, 2024. URL https://proceedings.mlr.press/v235/akyurek24a.html.

[8] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10041–10071. PMLR, 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

[9] Bernard Widrow, Marcian E Hoff, et al. Adaptive switching circuits. In *IRE WESCON convention record*, volume 4, pages 96–104. New York, 1960.

[10] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems 37*, pages 115491–115522, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d13a3eae72366e61dfdc7eea82eeb685-Abstract-Conference.html.

[11] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

[12] Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

[13] Aakash Lahoti, Kevin Y. Li, Berlin Chen, Caitlin Wang, Aviv Bick, J. Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=HwCvaJOiCj.

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

[15] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. Transformer quality in linear time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9099–9117. PMLR, 2022. URL https://proceedings.mlr.press/v162/hua22a.html.

[16] 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*, abs/2307.08621, 2023. URL https://arxiv.org/abs/2307.08621.

[17] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 56501–56523. PMLR, 2024. URL https://proceedings.mlr.press/v235/yang24ab.html.

[18] Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. The dual form of neural networks revisited: Connecting test time predictions to training patterns via spotlights of attention. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9639–9659. PMLR, 2022. URL https://proceedings.mlr.press/v162/irie22a.html.

[19] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 57503–57522. PMLR, 2025. URL https://proceedings.mlr.press/v267/sun25h.html.

[20] Riccardo Grazzi, Julien Siems, Arber Zela, Jörg K. H. Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=UvTo3tVBk2.

[21] Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners. In *International Conference on Learning Representations*, volume 2025, pages 95419–95434, 2025.

[22] Thierry Joffrain, Tze Meng Low, Enrique S. Quintana-Ortí, Robert A. van de Geijn, and Field G. Van Zee. Accumulating householder transformations, revisited. *ACM Trans. Math. Softw.*, 32:169–179, 2006. URL https://api.semanticscholar.org/CorpusID:15723171.

[23] Philippe Tillet, Hsiang-Tsung Kung, and David D. Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*. ACM, 2019.

[24] Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas,

and Caglar Gulcehre. Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models, 2024. URL https://arxiv.org/abs/2402.19427.

[25] Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=bIlnpVM4bc.

[26] Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *ArXiv preprint*, abs/2406.17557, 2024. URL https://arxiv.org/abs/2406.17557.

[27] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Byj72udxe.

[28] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 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, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144.

[29] Yonatan Bisk, Rowan Zellers, Ronan LeBras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6239.

[30] 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 Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

[31] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6399.

[32] 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*, abs/1803.05457, 2018. URL https://arxiv.org/abs/1803.05457.

[33] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260.

[34] 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, 2019. Association for

Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454.

[35] 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, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300.

[36] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *ArXiv preprint*, abs/2404.06654, 2024. URL https://arxiv.org/abs/2404.06654.

[37] Simran Arora, Aman Timalsina, Aaryan Singhal, Benjamin Spector, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Ré. Just read twice: closing the recall gap for recurrent language models. In *Proceedings of the 2nd Efficient Systems for Foundation Models Workshop at the International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, 2024.

[38] NVIDIA. NVIDIA Nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025.

[39] NVIDIA. Nemotron 3 Nano: Open, efficient mixture-of-experts hybrid Mamba-Transformer model for agentic reasoning. *arXiv preprint arXiv:2512.20848*, 2025.

[40] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

[41] Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=Ai8Hw3AXqks.

[42] Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Çaglar Gülçehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 26670–26698. PMLR, 2023. URL https://proceedings.mlr.press/v202/orvieto23a.html.

[43] Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. 2023.

[44] Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *ArXiv preprint*, abs/2404.07904, 2024. URL https://arxiv.org/abs/2404.07904.

[45] E. Gardner. The space of interactions in neural network models. *Journal of Physics A*, 21: 257–270, 1988. URL https://api.semanticscholar.org/CorpusID:15378089.

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

[47] Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. In Marc'Aurelio Ranzato,

Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 7703–7717, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/3f9e3767ef3b10a0de4c256d7ef9805d-Abstract.html.

[48] Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. OpenCeres: When open information extraction meets the semi-structured web. 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 3047–3056, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1309. URL https://aclanthology.org/N19-1309.

[49] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes, 2023. URL https://arxiv.org/abs/2304.09433.

[50] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2124. URL https://aclanthology.org/P18-2124.

[51] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147.

[52] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246.

[53] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026.

[54] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, 2024.

[55] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

[56] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[57] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[58] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[59] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021.

[60] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

[61] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, 2017.

[62] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9 (8):1735–1780, 1997.

[63] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *ArXiv preprint*, abs/2405.04517, 2024. URL https://arxiv.org/abs/2405.04517.

[64] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi 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 Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the transformer era. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.936. URL https://aclanthology.org/2023.findings-emnlp.936.

[65] Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, Kranthi Kiran GV, Jan Kocoń, Bartłomiej Koptyra, Satyapriya Krishna, Ronald McClelland Jr., Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Stanisław Woźniak, Ruichong Zhang, Bingchen Zhao, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. Eagle and Finch: RWKV with Matrix-Valued States and Dynamic Recurrence, 2024. URL https://arxiv.org/abs/2404.05892.

[66] Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. RWKV-7 "Goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

[67] Pingwei Sun, Yuxuan Hu, Jianchao Tan, Xue Wang, Jiaqi Zhang, Yifan Lu, Yerui Sun, Yuchen Xie, and Xunliang Cai. FG^2-GDN: Enhancing long-context gated delta networks with doubly fine-grained control. *arXiv preprint arXiv:2604.19021*, 2026.

[68] Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Improving state-tracking in linear rnns via householder products. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=SoRiaijTGr.

[69] 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.

[70] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *ArXiv preprint*, abs/2403.19887, 2024. URL https://arxiv.org/abs/2403.19887.

[71] William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=QZgo9JZpLq.

Appendix

A Chunkwise derivation for Gated DeltaNet-2

This appendix gives the exact chunkwise form used in Section 3.3. We work inside one chunk of length C and write $\mathbf{S}_0$ for the state at the start of the chunk. All vectors are for a single head. In particular, $\boldsymbol{k}_r, \boldsymbol{e}_r, \boldsymbol{q}_r, \boldsymbol{b}_r, \boldsymbol{\alpha}_r, \boldsymbol{\gamma}_r \in \mathbb{R}^{d_k}$, $\boldsymbol{v}_r, \boldsymbol{z}_r, \boldsymbol{o}_r, \mathbf{w}_r \in \mathbb{R}^{d_v}$, and $\mathbf{S}_r \in \mathbb{R}^{d_k \times d_v}$. The chunk index is suppressed.

The Gated DeltaNet-2 recurrence is

$$\mathbf{S}_r = \left(\mathbf{I} - \boldsymbol{k}_r \boldsymbol{e}_r^\top\right) \mathrm{Diag}(\boldsymbol{\alpha}_r)\mathbf{S}_{r-1} + \boldsymbol{k}_r \boldsymbol{z}_r^\top, \qquad \boldsymbol{e}_r = \boldsymbol{b}_r \odot \boldsymbol{k}_r, \qquad \boldsymbol{z}_r = \mathbf{w}_r \odot \boldsymbol{v}_r. \tag{30}$$

Let

$$\boldsymbol{G}_r = \sum_{i=1}^{r} \boldsymbol{g}_i, \qquad \boldsymbol{\gamma}_r = \exp(\boldsymbol{G}_r), \qquad \boldsymbol{\gamma}_0 = \mathbf{1}_{d_k}, \qquad \boldsymbol{\alpha}_r = \exp(\boldsymbol{g}_r). \tag{31}$$

All exponentials, products, and ratios involving $\boldsymbol{\gamma}$ are elementwise over the key channel axis.

A.1 Decay-normalized recurrence

Define a normalized state $\widehat{\mathbf{S}}_r$ by

$$\mathbf{S}_r = \mathrm{Diag}(\boldsymbol{\gamma}_r)\widehat{\mathbf{S}}_r. \tag{32}$$

Because $\boldsymbol{\gamma}_0 = \mathbf{1}_{d_k}$, the normalized initial state is also $\widehat{\mathbf{S}}_0 = \mathbf{S}_0$. Substituting Eq. 32 into Eq. 30 and using $\boldsymbol{\gamma}_r = \boldsymbol{\alpha}_r \odot \boldsymbol{\gamma}_{r-1}$ gives

$$\widehat{\mathbf{S}}_r = \left(\mathbf{I} - \bar{\boldsymbol{k}}_r \bar{\boldsymbol{e}}_r^\top\right)\widehat{\mathbf{S}}_{r-1} + \bar{\boldsymbol{k}}_r \boldsymbol{z}_r^\top, \qquad \bar{\boldsymbol{k}}_r = \boldsymbol{\gamma}_r^{-1} \odot \boldsymbol{k}_r, \qquad \bar{\boldsymbol{e}}_r = \boldsymbol{\gamma}_r \odot \boldsymbol{e}_r. \tag{33}$$

The channel-wise decay has disappeared from the recurrence. It is now carried by the left and right factors of each rank-one edit.

Let $\mathbf{K}$, $\mathbf{V}$, $\mathbf{B}$, and $\mathbf{W}$ contain rows $\boldsymbol{k}_r^\top$, $\boldsymbol{v}_r^\top$, $\boldsymbol{b}_r^\top$, and $\mathbf{w}_r^\top$, respectively. For compact matrix notation, let $\boldsymbol{\gamma}$ contain rows $\boldsymbol{\gamma}_r^\top$. Let $\bar{\mathbf{K}}$, $\bar{\mathbf{E}}$, and $\mathbf{Z}$ contain rows $\bar{\boldsymbol{k}}_r^\top$, $\bar{\boldsymbol{e}}_r^\top$, and $\boldsymbol{z}_r^\top$. Equivalently,

$$\bar{\mathbf{K}} = \boldsymbol{\gamma}^{-1} \odot \mathbf{K}, \qquad \bar{\mathbf{E}} = \boldsymbol{\gamma} \odot (\mathbf{B} \odot \mathbf{K}), \qquad \mathbf{Z} = \mathbf{W} \odot \mathbf{V}. \tag{34}$$

Define

$$\mathbf{T} = \mathrm{tril}(\bar{\mathbf{E}}\bar{\mathbf{K}}^\top, -1), \qquad \mathbf{A} = (\mathbf{I} + \mathbf{T})^{-1}, \qquad \mathbf{Y} = \mathbf{A}\bar{\mathbf{E}}, \qquad \mathbf{U} = \mathbf{A}\mathbf{Z}. \tag{35}$$

Since $\mathbf{T}$ is strictly lower triangular, $\mathbf{A}$ is lower triangular with unit diagonal and is obtained by forward substitution.

A.2 Compact state formula

Define

$$\mathbf{R} = \mathbf{U} - \mathbf{Y}\mathbf{S}_0. \tag{36}$$

Let row r of $\mathbf{R}$ be $\boldsymbol{\rho}_r^\top$, where $\boldsymbol{\rho}_r \in \mathbb{R}^{d_v}$. Then the normalized state after any prefix of the chunk is

$$\widehat{\mathbf{S}}_r = \mathbf{S}_0 + \bar{\mathbf{K}}_{\le r}^\top \mathbf{R}_{\le r}, \tag{37}$$

where $\bar{\mathbf{K}}_{\le r}$ and $\mathbf{R}_{\le r}$ denote the first r rows.

To prove Eq. 37, write the rank-one increment at step r as $\bar{\boldsymbol{k}}_r \boldsymbol{\rho}_r^\top$. From Eq. 33, the residual row is

$$\boldsymbol{\rho}_r^\top = \boldsymbol{z}_r^\top - \bar{\boldsymbol{e}}_r^\top \widehat{\mathbf{S}}_{r-1}. \tag{38}$$

Equivalently, in column-vector form, $\boldsymbol{\rho}_r = \boldsymbol{z}_r - \widehat{\mathbf{S}}_{r-1}^\top \bar{\boldsymbol{e}}_r$. Using the induction hypothesis $\widehat{\mathbf{S}}_{r-1} = \mathbf{S}_0 + \sum_{s<r} \bar{\boldsymbol{k}}_s \boldsymbol{\rho}_s^\top$ gives

$$\boldsymbol{\rho}_r^\top = \boldsymbol{z}_r^\top - \bar{\boldsymbol{e}}_r^\top \mathbf{S}_0 - \sum_{s<r} \bar{\boldsymbol{e}}_r^\top \bar{\boldsymbol{k}}_s \, \boldsymbol{\rho}_s^\top. \tag{39}$$

Since $\mathbf{T}_{rs} = \bar{\boldsymbol{e}}_r^\top \bar{\boldsymbol{k}}_s$ for $s < r$ and $\mathbf{T}_{rs} = 0$ otherwise, stacking these residual rows over the chunk yields

$$(\mathbf{I} + \mathbf{T})\mathbf{R} = \mathbf{Z} - \bar{\mathbf{E}}\mathbf{S}_0. \tag{40}$$

Multiplying by $\mathbf{A}$ gives $\mathbf{R} = \mathbf{A}\mathbf{Z} - \mathbf{A}\bar{\mathbf{E}}\mathbf{S}_0 = \mathbf{U} - \mathbf{Y}\mathbf{S}_0$, which is Eq. 36. Substituting the resulting increments into the normalized recurrence proves Eq. 37.

Multiplying Eq. 37 by $\mathrm{Diag}(\boldsymbol{\gamma}_C)$ gives the end-of-chunk state

$$\mathbf{S}_C = \mathrm{Diag}(\boldsymbol{\gamma}_C)\mathbf{S}_0 + \mathbf{K}_{\mathrm{tail}}^\top\big(\mathbf{U} - \mathbf{Y}\mathbf{S}_0\big), \tag{41}$$

where row r of $\mathbf{K}_{\mathrm{tail}}$ is

$$(\mathbf{K}_{\mathrm{tail}})_{r,:} = \big((\boldsymbol{\gamma}_C/\boldsymbol{\gamma}_r) \odot \boldsymbol{k}_r\big)^\top. \tag{42}$$

This is Eq. 23 in the main text.

A.3 Compact output formula

The output at token r is the column vector $\boldsymbol{o}_r = \mathbf{S}_r^\top \boldsymbol{q}_r$. It is convenient to write the corresponding row vector. Using Eq. 37,

$$\boldsymbol{o}_r^\top = (\boldsymbol{\gamma}_r \odot \boldsymbol{q}_r)^\top \mathbf{S}_0 + \sum_{s \le r} \big[\boldsymbol{q}_r^\top \mathrm{Diag}(\boldsymbol{\gamma}_r/\boldsymbol{\gamma}_s)\boldsymbol{k}_s\big]\boldsymbol{\rho}_s^\top. \tag{43}$$

Define $\mathbf{Q}_\gamma$ by row $(\mathbf{Q}_\gamma)_{r,:} = (\boldsymbol{\gamma}_r \odot \boldsymbol{q}_r)^\top$ and define the causal score matrix

$$(\mathbf{A}_{qk})_{rs} = \mathbf{1}_{r \ge s} \boldsymbol{q}_r^\top \mathrm{Diag}(\boldsymbol{\gamma}_r/\boldsymbol{\gamma}_s)\boldsymbol{k}_s. \tag{44}$$

Let $\mathbf{O}$ contain rows $\boldsymbol{o}_r^\top$. Stacking Eq. 43 over the chunk gives

$$\mathbf{O} = \mathbf{Q}_\gamma \mathbf{S}_0 + \mathbf{A}_{qk}\big(\mathbf{U} - \mathbf{Y}\mathbf{S}_0\big), \tag{45}$$

which is Eq. 24 in the main text.

A.4 Row recurrences

The matrices $\mathbf{Y}$ and $\mathbf{U}$ can also be written row by row. Let row r of $\mathbf{Y}$ be $\boldsymbol{y}_r^\top$ and row r of $\mathbf{U}$ be $\mathbf{u}_r^\top$. Since $(\mathbf{I} + \mathbf{T})\mathbf{Y} = \bar{\mathbf{E}}$ and $(\mathbf{I} + \mathbf{T})\mathbf{U} = \mathbf{Z}$,

$$\boldsymbol{y}_r^\top = \bar{\boldsymbol{e}}_r^\top - \sum_{s<r} \bar{\boldsymbol{e}}_r^\top \bar{\boldsymbol{k}}_s \, \boldsymbol{y}_s^\top, \tag{46}$$

$$\mathbf{u}_r^\top = \boldsymbol{z}_r^\top - \sum_{s<r} \bar{\boldsymbol{e}}_r^\top \bar{\boldsymbol{k}}_s \, \mathbf{u}_s^\top. \tag{47}$$

Both auxiliaries solve the same lower triangular system with different right-hand sides. This is why the same WY inverse can be shared by the erase-side and write-side computations.

A.5 Tied-gate reductions

If $\boldsymbol{b}_r = \beta_r \mathbf{1}_{d_k}$ and $\mathbf{w}_r = \beta_r \mathbf{1}_{d_v}$, then $\boldsymbol{e}_r = \beta_r \boldsymbol{k}_r$ and $\boldsymbol{z}_r = \beta_r \boldsymbol{v}_r$. Equation 30 becomes the KDA update. If the decay is also tied as $\boldsymbol{\alpha}_r = \alpha_r \mathbf{1}_{d_k}$, the recurrence becomes Gated DeltaNet. Thus KDA and Gated DeltaNet are recovered by tying the channel gates rather than by changing the algorithm.

The same reduction holds for the chunkwise form. Under the KDA tying, the definitions in Eq. 34 give

$$\bar{\boldsymbol{e}}_r = \boldsymbol{\gamma}_r \odot (\beta_r \boldsymbol{k}_r) = \beta_r(\boldsymbol{\gamma}_r \odot \boldsymbol{k}_r) = \beta_r(\boldsymbol{\gamma}_r \odot \boldsymbol{\gamma}_r) \odot \bar{\boldsymbol{k}}_r, \qquad \boldsymbol{z}_r = \beta_r \boldsymbol{v}_r. \tag{48}$$

Thus $\mathbf{Z}$ becomes a scalar row scaling of $\mathbf{V}$, while $\bar{\mathbf{E}}$ becomes the KDA decay-normalized erase factor. In general, $\bar{\mathbf{E}}$ is not a scalar row scaling of $\bar{\mathbf{K}}$ when the decay is channel-wise, since each key channel carries its own factor from $\boldsymbol{\gamma}_r$. Equations 35, 41, and 45 therefore reduce to the KDA chunk equations after substituting the tied factors in Eq. 48. If the decay is further tied as in Gated DeltaNet, then $\boldsymbol{\gamma}_r = \gamma_r \mathbf{1}_{d_k}$ and the erase factor simplifies to $\bar{\boldsymbol{e}}_r = \beta_r \gamma_r^2 \bar{\boldsymbol{k}}_r$, recovering the scalar-decay chunkwise form.

At the gradient level, if a thin wrapper sets $\boldsymbol{b}_r = \beta_r \mathbf{1}_{d_k}$ and $\mathbf{w}_r = \beta_r \mathbf{1}_{d_v}$, the scalar gradient is

$$\frac{\partial \mathcal{L}}{\partial \beta_r} = \left\langle \frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_r}, \mathbf{1}_{d_k} \right\rangle + \left\langle \frac{\partial \mathcal{L}}{\partial \mathbf{w}_r}, \mathbf{1}_{d_v} \right\rangle. \tag{49}$$

B Backward derivation

We derive the vector-Jacobian products for one chunk using the notation of Appendix A. Let upstream gradients be $\mathrm{d}\mathbf{O}$ for the chunk output and $\mathrm{d}\mathbf{S}_C$ for the end-of-chunk state. The forward equations are

$$\mathbf{R} = \mathbf{U} - \mathbf{Y}\mathbf{S}_0, \tag{50}$$
$$\mathbf{O} = \mathbf{Q}_\gamma \mathbf{S}_0 + \mathbf{A}_{qk}\mathbf{R}, \tag{51}$$
$$\mathbf{S}_C = \mathrm{Diag}(\boldsymbol{\gamma}_C)\mathbf{S}_0 + \mathbf{K}_{\mathrm{tail}}^\top \mathbf{R}, \tag{52}$$
$$\mathbf{U} = \mathbf{A}\mathbf{Z}, \qquad \mathbf{Y} = \mathbf{A}\bar{\mathbf{E}}, \qquad \mathbf{A} = (\mathbf{I} + \mathbf{T})^{-1}, \qquad \mathbf{T} = \mathrm{tril}(\bar{\mathbf{E}}\bar{\mathbf{K}}^\top, -1). \tag{53}$$

B.1 Output and state paths

From Eq. 51,

$$\mathrm{d}\mathbf{A}_{qk} \mathrel{+}= \mathrm{d}\mathbf{O}\,\mathbf{R}^\top, \tag{54}$$
$$\mathrm{d}\mathbf{R} \mathrel{+}= \mathbf{A}_{qk}^\top \mathrm{d}\mathbf{O}, \tag{55}$$
$$\mathrm{d}\mathbf{Q}_\gamma \mathrel{+}= \mathrm{d}\mathbf{O}\,\mathbf{S}_0^\top, \tag{56}$$
$$\mathrm{d}\mathbf{S}_0 \mathrel{+}= \mathbf{Q}_\gamma^\top \mathrm{d}\mathbf{O}. \tag{57}$$

The causal mask is applied to $\mathrm{d}\mathbf{A}_{qk}$.

From Eq. 52,

$$\mathrm{d}\mathbf{R} \mathrel{+}= \mathbf{K}_{\mathrm{tail}} \mathrm{d}\mathbf{S}_C, \tag{58}$$
$$\mathrm{d}\mathbf{K}_{\mathrm{tail}} \mathrel{+}= \mathbf{R}\,\mathrm{d}\mathbf{S}_C^\top, \tag{59}$$
$$\mathrm{d}\mathbf{S}_0 \mathrel{+}= \mathrm{Diag}(\boldsymbol{\gamma}_C)\mathrm{d}\mathbf{S}_C, \tag{60}$$
$$\mathrm{d}\boldsymbol{\gamma}_C \mathrel{+}= \mathrm{rowsum}\big(\mathrm{d}\mathbf{S}_C \odot \mathbf{S}_0\big). \tag{61}$$

Here rowsum sums over the value dimension and returns a d_k-dimensional vector.

The residual relation in Eq. 50 gives

$$\mathrm{d}\mathbf{U} \mathrel{+}= \mathrm{d}\mathbf{R}, \tag{62}$$
$$\mathrm{d}\mathbf{Y} \mathrel{+}= -\mathrm{d}\mathbf{R}\,\mathbf{S}_0^\top, \tag{63}$$
$$\mathrm{d}\mathbf{S}_0 \mathrel{+}= -\mathbf{Y}^\top \mathrm{d}\mathbf{R}. \tag{64}$$

B.2 Gate-aware WY inverse path

The two auxiliary products yield

$$\mathrm{d}\mathbf{A} \mathrel{+}= \mathrm{d}\mathbf{U}\,\mathbf{Z}^\top, \qquad \mathrm{d}\mathbf{Z} \mathrel{+}= \mathbf{A}^\top \mathrm{d}\mathbf{U}, \tag{65}$$
$$\mathrm{d}\mathbf{A} \mathrel{+}= \mathrm{d}\mathbf{Y}\,\bar{\mathbf{E}}^\top, \qquad \mathrm{d}\bar{\mathbf{E}} \mathrel{+}= \mathbf{A}^\top \mathrm{d}\mathbf{Y}. \tag{66}$$

Equations 65 and 66 are the gate-aware accumulation emphasized in Section 3.4. Since $\mathbf{Z} = \mathbf{W} \odot \mathbf{V}$ and $\bar{\mathbf{E}} = \boldsymbol{\gamma} \odot (\mathbf{B} \odot \mathbf{K})$, the gates must appear inside the products that accumulate $\mathrm{d}\mathbf{A}$. A scalar post-scale is correct only in the tied-gate case.

For the inverse,

$$\mathrm{d}\mathbf{T} = -\,\mathrm{tril}\big(\mathbf{A}^\top \mathrm{d}\mathbf{A}\,\mathbf{A}^\top, -1\big). \tag{67}$$

The construction of $\mathbf{T}$ gives

$$\mathrm{d}\bar{\mathbf{E}} \mathrel{+}= \mathrm{d}\mathbf{T}\,\bar{\mathbf{K}}, \tag{68}$$
$$\mathrm{d}\bar{\mathbf{K}} \mathrel{+}= \mathrm{d}\mathbf{T}^\top \bar{\mathbf{E}}. \tag{69}$$

Only the strictly lower triangular part of $\mathrm{d}\mathbf{T}$ is used.

B.3 Scores and tail keys

The score matrix satisfies $\mathbf{A}_{qk} = \mathrm{tril}(\mathbf{Q}_\gamma \bar{\mathbf{K}}^\top)$. Therefore

$$\mathrm{d}\mathbf{Q}_\gamma \mathrel{+}= \mathrm{d}\mathbf{A}_{qk}\, \bar{\mathbf{K}}, \tag{70}$$

$$\mathrm{d}\bar{\mathbf{K}} \mathrel{+}= \mathrm{d}\mathbf{A}_{qk}^\top \mathbf{Q}_\gamma. \tag{71}$$

The tail key matrix satisfies $(\mathbf{K}_{\mathrm{tail}})_r = \boldsymbol{\gamma}_C \odot \bar{\boldsymbol{k}}_r$. Hence

$$\mathrm{d}\bar{\mathbf{K}} \mathrel{+}= \mathrm{d}\mathbf{K}_{\mathrm{tail}} \odot \boldsymbol{\gamma}_C, \tag{72}$$

$$\mathrm{d}\boldsymbol{\gamma}_C \mathrel{+}= \sum_{r=1}^{C} \mathrm{d}(\mathbf{K}_{\mathrm{tail}})_r \odot \bar{\boldsymbol{k}}_r. \tag{73}$$

B.4 Elementwise gates and cumulative decay

The write-side relation $\mathbf{Z} = \mathbf{W} \odot \mathbf{V}$ gives

$$\mathrm{d}\mathbf{W} \mathrel{+}= \mathrm{d}\mathbf{Z} \odot \mathbf{V}, \qquad \mathrm{d}\mathbf{V} \mathrel{+}= \mathrm{d}\mathbf{Z} \odot \mathbf{W}. \tag{74}$$

The erase-side relation $\bar{\mathbf{E}} = \boldsymbol{\gamma} \odot (\mathbf{B} \odot \mathbf{K})$ gives

$$\mathrm{d}\mathbf{B} \mathrel{+}= \mathrm{d}\bar{\mathbf{E}} \odot \boldsymbol{\gamma} \odot \mathbf{K}, \tag{75}$$

$$\mathrm{d}\mathbf{K} \mathrel{+}= \mathrm{d}\bar{\mathbf{E}} \odot \boldsymbol{\gamma} \odot \mathbf{B}, \tag{76}$$

$$\mathrm{d}\boldsymbol{\gamma} \mathrel{+}= \mathrm{d}\bar{\mathbf{E}} \odot \mathbf{B} \odot \mathbf{K}. \tag{77}$$

The normalized keys and queries are

$$\bar{\mathbf{K}} = \boldsymbol{\gamma}^{-1} \odot \mathbf{K}, \qquad \mathbf{Q}_\gamma = \boldsymbol{\gamma} \odot \mathbf{Q}. \tag{78}$$

Their vector-Jacobian products are

$$\mathrm{d}\mathbf{K} \mathrel{+}= \mathrm{d}\bar{\mathbf{K}} \odot \boldsymbol{\gamma}^{-1}, \tag{79}$$

$$\mathrm{d}\boldsymbol{\gamma} \mathrel{+}= -\mathrm{d}\bar{\mathbf{K}} \odot \mathbf{K} \odot \boldsymbol{\gamma}^{-2}, \tag{80}$$

$$\mathrm{d}\mathbf{Q} \mathrel{+}= \mathrm{d}\mathbf{Q}_\gamma \odot \boldsymbol{\gamma}, \tag{81}$$

$$\mathrm{d}\boldsymbol{\gamma} \mathrel{+}= \mathrm{d}\mathbf{Q}_\gamma \odot \mathbf{Q}. \tag{82}$$

Finally, $\boldsymbol{\gamma}_r = \exp(\boldsymbol{G}_r)$ and $\boldsymbol{G}_r = \sum_{i \le r} \boldsymbol{g}_i$. Therefore

$$\mathrm{d}\boldsymbol{G}_r = \mathrm{d}\boldsymbol{\gamma}_r \odot \boldsymbol{\gamma}_r, \qquad \mathrm{d}\boldsymbol{g}_i = \sum_{r \ge i} \mathrm{d}\boldsymbol{G}_r. \tag{83}$$

In implementation this is a reverse cumulative sum over the chunk.

B.5 Why scalar post-scaling is invalid

In KDA, a scalar β_r multiplies both the value right hand side and the erase right hand side. For one row, the contribution to $\mathrm{d}\mathbf{A}$ from the write auxiliary can be factored as

$$\mathrm{d}\mathbf{u}_r \, (\beta_s \boldsymbol{v}_s)^\top = \beta_s \, \mathrm{d}\mathbf{u}_r \boldsymbol{v}_s^\top. \tag{84}$$

The scalar factor can be applied after the dot product. Gated DeltaNet-2 replaces $\beta_s \boldsymbol{v}_s$ by $\mathbf{w}_s \odot \boldsymbol{v}_s$. Since $\mathbf{w}_s$ is a different diagonal operator for every row, there is no row scalar or column scalar that can recover

$$\mathrm{d}\mathbf{u}_r \, (\mathbf{w}_s \odot \boldsymbol{v}_s)^\top \tag{85}$$

from $\mathrm{d}\mathbf{u}_r \boldsymbol{v}_s^\top$. The erase side has the same issue with $\boldsymbol{b}_s \odot \boldsymbol{k}_s$. The gates must be baked into the dot products in Eq. 65 and Eq. 66.

C Layer and kernel implementation

This appendix records the implementation choices needed to reproduce Gated DeltaNet-2. The main text keeps the Triton details brief. Here we describe the computation at the level of kernels and tensor shapes.

C.1 Layer parameterization

The layer computes short-convolutional projections for $\boldsymbol{q}$, $\boldsymbol{k}$, and $\boldsymbol{v}$, followed by head reshaping. The erase and write gates are produced by independent projections,

$$\boldsymbol{b} = \sigma(\mathrm{Proj}_b(\boldsymbol{x})), \qquad \mathbf{w} = \sigma(\mathrm{Proj}_w(\boldsymbol{x})). \tag{86}$$

The erase projection has shape $d_{\text{model}} \to Hd_k$. The write projection has shape $d_{\text{model}} \to H_v d_v$. If grouped value attention is used with $H_v > H$, the key-side tensors $\boldsymbol{q}$, $\boldsymbol{k}$, the log-decay tensor $\boldsymbol{g}$, and $\boldsymbol{b}$ are repeated across the value-head group, while $\boldsymbol{v}$ and $\mathbf{w}$ already live on the value-head axis.

The log-decay is computed outside the kernel in fp32,

$$\boldsymbol{g}_t = -\exp(\mathbf{a}) \odot \mathrm{softplus}(\mathrm{Proj}_f(\boldsymbol{x}_t) + \boldsymbol{\delta}). \tag{87}$$

The vector $\mathbf{a}$ is stored per key head and broadcast across the d_k channels of that head. The bias $\boldsymbol{\delta}$ is stored per key channel. The kernel consumes $\boldsymbol{g}_t$ directly and forms the local cumulative sums in Eq. 31.

If negative eigenvalues are enabled, only the erase gate is scaled by 2. This changes $\boldsymbol{b}_t \in [0,1]^{d_k}$ into $\boldsymbol{b}_t \in [0,2]^{d_k}$. The write gate remains in $[0,1]^{d_v}$.

C.2 Forward kernels

The chunk size is fixed to $C = 64$. Each chunk is processed by the following steps.

Intra-chunk products The first kernel forms the causal score matrix $\mathbf{A}_{qk}$ and the strictly lower matrix $\mathbf{T}$. The Gated DeltaNet-2 specific computation is the row factor of $\mathbf{T}$,

$$T_{rs} = \bar{\boldsymbol{e}}_r^\top \bar{\boldsymbol{k}}_s = (\boldsymbol{\gamma}_r \odot \boldsymbol{b}_r \odot \boldsymbol{k}_r)^\top (\boldsymbol{\gamma}_s^{-1} \odot \boldsymbol{k}_s) \qquad s < r. \tag{88}$$

Thus the erase gate is multiplied into the key tile before the dot product. The score matrix $\mathbf{A}_{qk}$ is unchanged apart from the same decay-normalized key factors.

WY solve The second kernel solves $\mathbf{A} = (\mathbf{I} + \mathbf{T})^{-1}$ by forward substitution. It then exposes the same lower triangular inverse to both right hand sides. This is the compact WY step used in Eq. 35.

Auxiliary construction The third kernel builds

$$\mathbf{U} = \mathbf{A}(\mathbf{W} \odot \mathbf{V}), \qquad \mathbf{Y} = \mathbf{A}\bar{\mathbf{E}}. \tag{89}$$

The implementation stores $\mathbf{Y}$ using the historical buffer name `w` because KDA used the same buffer for its erase-side auxiliary; this buffer is not the write-gate matrix $\mathbf{W}$. The mathematical role is $\mathbf{Y}$ throughout this paper.

State and output The inter-chunk state recurrence consumes $\mathbf{K}_{\text{tail}}$, $\mathbf{Y}$, and $\mathbf{U}$, and applies Eq. 41. The output kernel consumes $\mathbf{Q}_\gamma$, $\mathbf{A}_{qk}$, and $\mathbf{R} = \mathbf{U} - \mathbf{Y}\mathbf{S}_0$, and applies Eq. 45. These two kernels do not depend on how the gate factors were produced, so they share the same matrix shapes as KDA.

C.3 Backward kernels

The backward pass mirrors Appendix B.

Output vector-Jacobian product The first backward kernel computes $\mathrm{d}\mathbf{A}_{qk}$ and the output-path contribution to $\mathrm{d}\mathbf{R}$ from Eq. 54 and Eq. 55. It is structure-equivalent to the KDA output vector-Jacobian product because it only sees $\mathbf{R}$.

State vector-Jacobian product The second backward kernel propagates $\mathrm{d}\mathbf{S}_C$ through Eq. 52. It is also structure-equivalent to the KDA state vector-Jacobian product because it consumes $\mathbf{K}_{\text{tail}}$, $\mathbf{Y}$, and $\mathbf{U}$ as already formed tensors.

Gate-aware WY vector-Jacobian product The third backward kernel implements Eq. 65, Eq. 66, and Eq. 67. This is the main Gated DeltaNet-2 specific kernel. It accumulates $\mathrm{d}\mathbf{A}$ with $\mathbf{Z}^\top = (\mathbf{W} \odot \mathbf{V})^\top$ and $\bar{\mathbf{E}}^\top = (\boldsymbol{\gamma} \odot \mathbf{B} \odot \mathbf{K})^\top$. It also emits the direct gradients

$$\mathrm{d}\mathbf{W} = \mathrm{d}\mathbf{Z} \odot \mathbf{V}, \qquad \mathrm{d}\mathbf{V} = \mathrm{d}\mathbf{Z} \odot \mathbf{W}, \qquad \mathrm{d}\mathbf{B} \mathrel{+}= \mathrm{d}\bar{\mathbf{E}} \odot \boldsymbol{\gamma} \odot \mathbf{K}. \tag{90}$$

The erase-gate gradient has shape $B \times T \times H \times d_k$. The write-gate gradient has shape $B \times T \times H_v \times d_v$.

Intra-chunk vector-Jacobian product The fourth backward kernel propagates through $\mathbf{A}_{qk}$ and $\mathbf{T}$. It adds the remaining contributions to $\mathrm{d}\mathbf{Q}$, $\mathrm{d}\mathbf{K}$, $\mathrm{d}\mathbf{B}$, and $\mathrm{d}\boldsymbol{g}$. The dependence on the cumulative decay is reduced by a reverse cumulative sum, as in Eq. 83.

C.4 Autotuning and hardware dispatch

The fused WY backward kernel uses the same matrix shapes as the forward solve, but it has a denser set of live accumulators because it emits gradients for $\mathbf{B}$ and $\mathbf{W}$. On Hopper GPUs we restrict the warp search for this kernel to two and four warps, since the eight-warp schedule can trigger a Triton WGMMA layout assertion for the 64×64 accumulator. On Ampere GPUs the full search space is retained. This restriction changes only the schedule, not the mathematical operation.

C.5 Recurrent decoding kernel

A forward-only recurrent kernel is provided for autoregressive decoding at short sequence lengths. It applies Eq. 30 token by token. The kernel keeps the state in fp32, multiplies it by $\exp(\boldsymbol{g}_t)$, reads the decayed state along $\boldsymbol{b}_t \odot \boldsymbol{k}_t$, writes $\mathbf{w}_t \odot \boldsymbol{v}_t$ along $\boldsymbol{k}_t$, and returns $\mathbf{S}_t^\top \boldsymbol{q}_t$. Training uses the chunk kernel.

C.6 Variable-length sequences

Packed variable-length batches are represented with cumulative sequence lengths. The chunk index construction resets the recurrent state at every sequence boundary. The same layout is used by the chunk forward, the chunk backward, and the recurrent decoding kernel. Padding is removed before the layer and restored after the output projection.

D Numerical details and verification

D.1 Decay precision

The decay gate in Eq. 87 is computed in explicit fp32 before entering the kernels. This is important because the local cumulative sum $\boldsymbol{G}_r = \sum_{i \le r} \boldsymbol{g}_i$ is a path-length-dependent quantity. A low precision mantissa can perturb long products of decays even when each tokenwise gate is small. The kernels therefore receive the log-decay tensor and only compute local cumulative sums and exponentials.

D.2 Query and key normalization

Queries and keys are L2-normalized per head before the recurrent update. With normalized keys, $\boldsymbol{k}_t \boldsymbol{k}_t^\top$ is a projector in the tied-gate limit. In Gated DeltaNet-2, the erase factor is asymmetric, but normalization still stabilizes the scale of both $\mathbf{A}_{qk}$ and $\mathbf{T}$. The backward applies the standard L2-normalization vector-Jacobian product.

D.3 State and accumulator dtypes

The recurrent state is stored in fp32 across chunks and during recurrent decoding. Matrix multiplication accumulators use fp32. The layer output is cast back to the model dtype at the kernel boundary. The WY auxiliaries may be stored in the model dtype after fp32 accumulation, since they are recomputed when the memory-saving training path is used.

D.4 WY solve precision

The triangular solve for $\mathbf{A} = (\mathbf{I} + \mathbf{T})^{-1}$ is the most precision-sensitive part of the chunk computation. Errors in this solve are propagated through dependent forward-substitution steps. The implementation therefore exposes an explicit precision flag for the solve. The conservative IEEE fp32 path is used when required by the hardware check, while the remaining matrix products can use the faster tensor-core path.

D.5 Initialization and output gate

All linear layers are initialized with Xavier uniform weights and gain $2^{-2.5}$. Biases are initialized to zero when present. After the attention computation, the output is passed through an RMSNorm and SiLU gate before the final output projection. These choices match the training recipe used for the Gated DeltaNet family and keep the early recurrent state magnitudes controlled.

D.6 Correctness checks

We verified the chunkwise forward against a tokenwise recurrent reference for random configurations covering different sequence lengths, head counts, key dimensions, value dimensions, initial states, packed layouts, and dtypes. We verified the backward against autograd through the recurrent reference. In fp64 reference tests, gradients for $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$, $\mathbf{B}$, $\mathbf{W}$, the log-decay, and the initial state agree to machine precision. In production fp32, differences are at the expected tensor-core accumulation noise level. In bfloat16, the error follows the bfloat16 mantissa.

E Experimental settings

E.1 Training

We evaluate Gated DeltaNet-2 against a Transformer baseline and recent recurrent architectures, including Mamba-2 [8], Gated DeltaNet [11], Kimi Delta Attention (KDA) [12], and Mamba-3 [13]. For each recurrent architecture, we train a recurrent-only model and a hybrid model. The hybrid model follows Section 3.5, using the same recurrent token mixer together with sliding-window attention (SWA) under the same residual block structure. For Mamba-3, we evaluate both SISO and MIMO variants. The Mamba-3 MIMO model uses rank $R = 4$.

For fair recurrent comparisons, we match both parameter count and main recurrent state size. Gated DeltaNet, KDA, and Gated DeltaNet-2 use $H = 16$ heads with $d_k = 128$ and $d_v = 128$, giving a per-layer recurrent state of

$$H d_k d_v = 16 \cdot 128 \cdot 128 = 262{,}144 \tag{91}$$

floats per batch element. Since $d_{\text{model}} = 2048$, this equals $128 d_{\text{model}}$. For Mamba-2 and Mamba-3, we use expansion factor 2 and head dimension 64, and set $d_{\text{state}} = 64$. Their main recurrent state size is therefore

$$(2d_{\text{model}}) d_{\text{state}} = 4096 \cdot 64 = 262{,}144. \tag{92}$$

Mamba-3 MIMO keeps the same main recurrent state size as the SISO variant while adding the rank-R MIMO parameterization.

Unless stated otherwise, all models have 1.3B parameters and are trained on 100B tokens sampled from FineWeb-Edu [26]. We use AdamW with peak learning rate 4×10^{-4}, weight decay 0.1, and gradient clipping at 1.0. The learning rate follows cosine annealing with a 1B-token warm-up. The global batch size is 0.5M tokens. The training sequence length is 4K tokens. Hybrid models use a 2K SWA window.

E.2 Evaluation

Language modeling and common-sense reasoning We use the evaluation suite commonly adopted for pretrained recurrent language models [43]. Language modeling quality is measured by perplexity on WikiText [Wiki. 27] and LAMBADA [LMB. 28]. For zero-shot transfer, we report LAMBADA

accuracy together with PIQA [29], HellaSwag [Hella. 30], WinoGrande [Wino. 31], ARC-Easy and ARC-Challenge [ARC-e and ARC-c 32], OpenBookQA [OBQA 33], Social IQa [SIQA 34], and BoolQ [35]. This mix covers next-token prediction, physical and social reasoning, commonsense completion, and elementary science QA.

In-context retrieval We evaluate retrieval in both controlled synthetic settings and real-data settings. For synthetic retrieval, we use Single Needle-In-A-Haystack (S-NIAH) and Multi-Key Needle-In-A-Haystack (MK-NIAH) tasks from RULER [36]. The S-NIAH suite contains three progressively harder cases. S-NIAH-1 is passkey retrieval, S-NIAH-2 asks for a numerical needle, and S-NIAH-3 asks for a word-based needle. We additionally evaluate MK-NIAH-1 where several distractor key-value pairs are present and the model must return the value associated with one requested key.

For real-world retrieval, we follow Arora et al. [37]. The suite includes SWDE [48] for structured relation extraction from HTML, FDA [49] for key-value retrieval from PDFs, and question-answering datasets including SQuAD [50], TriviaQA [51], DROP [52], and Natural Questions [53].

E.3 Large-scale global-attention hybrid

The scaling experiment in Table 5 uses a Nemotron-3 Nano-style hybrid latent-MoE architecture [38, 39], in which recurrent mixers and MoE layers are interleaved with a small number of full global-attention layers. Each model has 31.6B total parameters and 3.2B active parameters and is trained for 300B tokens. The global-attention layers and their placement, the recurrent-state size, the active parameter count, and the training recipe are identical across variants, and only the recurrent mixer is changed among Mamba-2, KDA, and Gated DeltaNet-2. Since the mixers differ slightly in parameter count, the FFN intermediate size is adjusted by a small amount so that all variants are iso-parameter.

The evaluation covers four categories. Knowledge is measured with MMLU-Pro [54] and MMLU-Str. [55]. Code is measured with HumanEval [56] and MBPP [57]. Math is measured with GSM8K [58] and MATH-500 [59, 60]. Commonsense understanding is measured with RACE [61], ARC-Challenge [32], HellaSwag [30], and WinoGrande [31]. Each category score is the mean of its benchmarks, and the overall score is the equal average of the four category scores.

F Extended related work

Section 5 positions Gated DeltaNet-2 among efficient sequence models. This appendix expands that discussion along the axis most relevant to our contribution, the use of several gates to control a recurrent memory, and relates the model to the associative-memory view of recent work.

Gated recurrent memories The principle of separating memory-control decisions into several gates dates back to the LSTM, whose elementwise input and forget gates act on a vector-valued cell state [62]. The mLSTM cell of xLSTM introduces a matrix memory, but still updates it by a gated covariance accumulation [63]. Modern linear attention and linear RNNs make the decay data-dependent and often vector-valued. Gated linear attention applies data-dependent vector gating to a matrix-state linear attention layer while retaining an additive key-value write [17], RWKV uses channel-wise decay and its RWKV-5 and RWKV-6 successors adopt matrix-valued states with dynamic vector decay [64, 65], HGRN2 expands the state with outer products under vector forget gates [44], Mamba-2 uses a scalar data-dependent decay [8], and KDA makes the decay of a delta-rule layer channel-wise [12]. None of the gated accumulators in this family performs the read-before-write delta correction of Gated DeltaNet-2,

$$\mathbf{S}_t = \bar{\mathbf{S}}_t + \boldsymbol{k}_t\big[(\mathbf{w}_t \odot \boldsymbol{v}_t) - \bar{\mathbf{S}}_t^\top(\boldsymbol{b}_t \odot \boldsymbol{k}_t)\big]^\top, \qquad \bar{\mathbf{S}}_t = \mathbf{D}_t\mathbf{S}_{t-1}. \tag{93}$$

Our contribution is therefore not the generic use of multiple or vector-valued gates. It is that the two gates act on different axes of a queryable matrix memory inside a delta-rule edit. The erase gate $\boldsymbol{b}_t$ controls the key-side direction along which an existing association is read and erased, while the write gate $\mathbf{w}_t$ independently controls which value channels are committed to memory.

Delta-rule and test-time-learning memories DeltaNet, Gated DeltaNet, and KDA, discussed in Section 2, all use a single scalar step size β_t for the delta edit. Two models relax this scalar. RWKV-7 generalizes the delta rule with vector-valued in-context learning rates and a removal key

that is decoupled from the replacement key, which makes it structurally the closest prior model to ours at the level of state evolution [66]. FG^2-GDN, concurrent with this work, makes β_t channel-wise, and its FG^2-GDN+ variant uses separate channel-wise scalings for keys and values [67]. Appendix G gives the formulation-level differences between these models and Gated Delta Rule-2 and reports a matched comparison. Beyond the step size, DeltaProduct composes several Householder transformations per token to improve state tracking [68], and Grazzi et al. [20] show that allowing negative eigenvalues in the state transition unlocks state tracking in linear RNNs. Gated DeltaNet-2 supports the corresponding expanded erase range, and Appendix I evaluates it on the DeltaProduct permutation tasks. The fast-weight interpretation of Section 3 follows the view of recurrent layers as online learners [2, 18, 19, 21].

The Miras view The Miras framework organizes sequence models through four design choices, the associative-memory architecture, the attentional-bias objective, the retention gate, and the memory-learning algorithm [69]. Under this taxonomy, Gated DeltaNet-2 uses a matrix associative memory, a regression-based delta objective, Eq. 13, a channel-wise diagonal retention gate $\mathbf{D}_t$, and a one-step online memory update. The retention-gate axis most directly describes how the previous memory is preserved or forgotten, which corresponds to $\mathbf{D}_t$ in Gated DeltaNet-2. Our model focuses on the finer question of how the subsequent active delta edit is factorized,

$$\boldsymbol{e}_t = \boldsymbol{b}_t \odot \boldsymbol{k}_t, \qquad \boldsymbol{z}_t = \mathbf{w}_t \odot \boldsymbol{v}_t, \qquad \mathbf{S}_t = \mathbf{D}_t\mathbf{S}_{t-1} + \boldsymbol{k}_t\big(\boldsymbol{z}_t - (\mathbf{D}_t\mathbf{S}_{t-1})^\top \boldsymbol{e}_t\big)^\top, \tag{94}$$

so that $\boldsymbol{b}_t$ independently controls the key-side read and erase direction while $\mathbf{w}_t$ controls the value-side commit target. This is a finer modeling choice within the Miras framework rather than an alternative framework.

Hybrid architectures Recurrent mixers are commonly combined with attention, either with sliding windows as in Griffin and Samba [24, 25] or with a small number of full global-attention layers as in Jamba and the Nemotron family [70, 38, 39]. Our 1.3B hybrids follow the first pattern, and the scaling experiment of Table 5 follows the second.

G Comparison with RWKV-7 and FG^2-GDN+

RWKV-7 [66] and the concurrent FG^2-GDN+ [67] both relax the single scalar step size of the delta rule, so it is natural to ask what Gated Delta Rule-2 adds. This appendix compares the three updates at the level of their equations and then reports a matched empirical comparison.

G.1 Formulation-level differences

FG^2-GDN+ FG^2-GDN+ predicts a channel-wise key step size $\boldsymbol{\beta}_t^k \in \mathbb{R}^{d_k}$ and a channel-wise value step size $\boldsymbol{\beta}_t^v \in \mathbb{R}^{d_v}$ and folds their square roots into the key and value,

$$\hat{\boldsymbol{k}}_t = \sqrt{\boldsymbol{\beta}_t^k} \odot \boldsymbol{k}_t, \qquad \tilde{\boldsymbol{v}}_t = \sqrt{\boldsymbol{\beta}_t^v} \odot \boldsymbol{v}_t, \qquad \mathbf{S}_t = \big(\mathbf{I} - \hat{\boldsymbol{k}}_t\hat{\boldsymbol{k}}_t^\top\big)\mathbf{D}_t\mathbf{S}_{t-1} + \hat{\boldsymbol{k}}_t\tilde{\boldsymbol{v}}_t^\top. \tag{95}$$

The same vector $\hat{\boldsymbol{k}}_t$ controls key-side erasure and key-side writing. Weakening erasure on key channel i therefore also weakens every write into row i of the state. The value gate $\boldsymbol{\beta}_t^v$ cannot compensate, because it scales value channels, the columns of $\mathbf{S}_t$, rather than key channels, the rows. Gated Delta Rule-2 instead uses

$$\mathbf{S}_t = \big(\mathbf{I} - \boldsymbol{k}_t(\boldsymbol{b}_t \odot \boldsymbol{k}_t)^\top\big)\mathbf{D}_t\mathbf{S}_{t-1} + \boldsymbol{k}_t(\mathbf{w}_t \odot \boldsymbol{v}_t)^\top, \tag{96}$$

where $\boldsymbol{b}_t$ affects only the read and erase direction, writing remains along the unmodified key $\boldsymbol{k}_t$, and $\mathbf{w}_t$ independently selects the committed value channels. Thus FG^2-GDN+ decouples the erase magnitude from value-side scaling, but retains a hard key-side coupling between erasing and writing that Gated Delta Rule-2 removes.

RWKV-7 RWKV-7 removes old content with a token-dependent channel-wise strength $\boldsymbol{a}_t$ and writes the new value along a replacement key that is a soft function of the same signal,

$$\tilde{\boldsymbol{k}}_t = \boldsymbol{k}_t \odot \mathrm{lerp}(\mathbf{1}, \boldsymbol{a}_t, \boldsymbol{\xi}) = \boldsymbol{k}_t \odot \big[\mathbf{1} + (\boldsymbol{a}_t - \mathbf{1}) \odot \boldsymbol{\xi}\big], \tag{97}$$

where $\boldsymbol{\xi}$, denoted α in RWKV-7, is a learned token-independent channel-wise replacement-rate booster. The same $\boldsymbol{a}_t$ therefore influences both how much content is removed and the key used to write new content, with $\boldsymbol{\xi}$ controlling the strength of this coupling. Setting $\boldsymbol{\xi}$ near zero on a channel removes the coupling, but only statically. RWKV-7 cannot choose the coupling independently per token, and it has no independently predicted gate that directly scales value channels in the state write. Gated Delta Rule-2 predicts independent token-dependent $\boldsymbol{b}_t$ and $\mathbf{w}_t$ by construction and removes the coupling across the two axes of its matrix memory.

G.2 Matched comparison

To compare the three updates empirically, we implemented the RWKV-7 and FG^2-GDN+ state updates in the same surrounding block as Gated DeltaNet-2 (Section 3.5) and matched the parameter count, the recurrent-state size, the 1.3B scale, the 100B-token FineWeb-Edu budget, the 4K training length, and the training recipe of Section 4. Table S.1 reports three aggregates. Avg. ppl averages WikiText and LAMBADA perplexity, Common Avg. follows the average of Table 2, and Recall Avg. averages the six real-world retrieval tasks of Table 4. Results are the mean and sample standard deviation over three independent pretraining seeds. The single-run Gated DeltaNet-2 checkpoint of Tables 2 to 4 corresponds to the first of these seeds and closely matches the three-seed mean.

Model	Avg. ppl ↓	Common Avg. ↑	Recall Avg. ↑
RWKV-7	14.27 ± 0.05	52.21 ± 0.14	28.54 ± 0.17
FG^2-GDN+	14.18 ± 0.04	52.38 ± 0.13	28.73 ± 0.16
Gated DeltaNet-2	$\mathbf{13.66 \pm 0.03}$	$\mathbf{53.10 \pm 0.10}$	$\mathbf{29.87 \pm 0.12}$

Table S.1: Matched comparison of RWKV-7, FG^2-GDN+, and Gated DeltaNet-2 in the recurrent-only 1.3B setting with 100B training tokens. All models share the same block design, parameter count, recurrent-state size, and training recipe. Values are mean ± sample standard deviation over three pretraining seeds.

Gated DeltaNet-2 outperforms both models on every aggregate, with margins well above the seed variation. Relative to FG^2-GDN+, it reduces average perplexity by 0.52 and improves the commonsense and recall averages by 0.72 and 1.14 points. Relative to RWKV-7, it reduces average perplexity by 0.61 and improves the two averages by 0.89 and 1.33 points. Since the three models differ only in how the delta edit is gated, these margins reflect the value of predicting the key-side erase direction and the value-side write target independently at every token.

H Additional ablations on the gate structure

Table 6 tests whether the erase and write gates need channel-wise structure by scalarizing either gate. This appendix adds two controls that isolate other aspects of Gated Delta Rule-2: the independence of the two gates and the incremental contribution of write selectivity. Both use the recurrent-only 1.3B setting and the metrics of Table 6. As in Table 6, the tables report single-run values; the shared-gate variant of Section H.1 was additionally trained with the same seed and data order as the Gated DeltaNet-2 checkpoint of Table 6, so that comparison is paired.

H.1 Tied versus decoupled channel-wise gates

Comparing Gated DeltaNet-2 with KDA changes two things at once. The delta gate becomes channel-wise, and the erase and write decisions become independent. The scalarized variants of Table 6 isolate the first change. To isolate the second, we train a shared-vector-gate variant that retains both gate projections and ties only their token-dependent outputs,

$$\boldsymbol{g}_t = \tfrac{1}{2}\big[\sigma(\mathbf{W}_b \boldsymbol{x}_t) + \sigma(\mathbf{W}_w \boldsymbol{x}_t)\big], \qquad \boldsymbol{b}_t = \mathbf{w}_t = \boldsymbol{g}_t, \tag{98}$$

with the recurrence

$$\mathbf{S}_t = \big(\mathbf{I} - \boldsymbol{k}_t(\boldsymbol{g}_t \odot \boldsymbol{k}_t)^\top\big)\mathbf{D}_t\mathbf{S}_{t-1} + \boldsymbol{k}_t(\boldsymbol{g}_t \odot \boldsymbol{v}_t)^\top. \tag{99}$$

Because $d_k = d_v$ in our configuration, the same channel-wise vector can be applied on both axes. The control preserves the parameter count, the gate projections, the recurrent-state size, the compute,

and the training recipe of Gated DeltaNet-2, and removes only the independence between erasing and writing.

Variant	Wiki. ppl ↓	LMB. ppl ↓	Common. avg ↑	S-NIAH-2 @4K ↑	S-NIAH-3 @2K ↑	MK-NIAH-1 @4K ↑	Recall avg ↑
Shared vector gate, $\boldsymbol{b}_t = \mathbf{w}_t$	16.07	11.49	52.86	91.8	85.2	35.1	29.52
Independent $\boldsymbol{b}_t$, $\mathbf{w}_t$ (Gated DeltaNet-2)	**15.90**	**11.41**	**53.11**	**93.0**	**89.8**	**37.8**	**29.88**

Table S.2: Tied versus decoupled channel-wise gates in the recurrent-only setting. The shared-vector-gate variant keeps both gate projections and ties their outputs, so it matches Gated DeltaNet-2 in parameters, state size, and compute.

Table S.2 shows that independent gates improve every metric, including 4.6 points on S-NIAH-3 and 2.7 points on MK-NIAH-1. Since vectorization, parameters, state size, compute, and training are fixed, the difference directly isolates the benefit of allowing key-side erasure and value-side writing to vary independently. The same control also rules out a capacity explanation of the gain over KDA. The shared-gate model retains both $\mathbf{W}_b$ and $\mathbf{W}_w$ and therefore has exactly the parameter count of Gated DeltaNet-2, yet it still trails on every reported metric.

H.2 Erase versus write selectivity

Table 6 indicates that key-side erase selectivity accounts for most of the improvement at the current scale and context length. To isolate the additional contribution of write selectivity, Table S.3 compares the variant that keeps $\boldsymbol{b}_t$ channel-wise but scalarizes $\mathbf{w}_t$ against the full model, using the single-run values of Table 6.

Variant	Common. avg ↑	S-NIAH-2 @4K ↑	S-NIAH-3 @2K ↑	MK-NIAH-1 @4K ↑	Recall avg ↑
Channel-wise $\boldsymbol{b}_t$, scalar $\mathbf{w}_t$	52.79	92.1	84.6	35.2	29.51
Channel-wise $\boldsymbol{b}_t$ and $\mathbf{w}_t$ (Gated DeltaNet-2)	**53.11**	**93.0**	**89.8**	**37.8**	**29.88**

Table S.3: Incremental contribution of write selectivity in the recurrent-only setting. Both rows keep the erase gate channel-wise; the first scalarizes the write gate as in Table 6.

Erase selectivity is the larger contributor at the 4K training length, but write selectivity provides substantial additional gains on interference-heavy retrieval, improving S-NIAH-3 by 5.2 points and MK-NIAH-1 by 2.6 points. The smaller gains on the commonsense and recall averages indicate that this effect is concentrated in settings where several associations compete for limited recurrent memory. The two gates address complementary bottlenecks. The erase gate determines which components of an existing association should be revised, while the write gate acts as admission control over which value channels enter memory. We therefore expect erase selectivity to remain important for overwrite-heavy retrieval, while write selectivity should become more valuable as longer and more information-dense contexts increase the pressure to avoid storing irrelevant content.

I State tracking on permutation composition

The main text focuses on retrieval, where fixed-state memory must manage interference. State tracking is a second capability that motivates recent recurrent models [20, 68, 13]. We therefore evaluate Gated DeltaNet-2 on the permutation-composition benchmark of DeltaProduct [68], which builds on the group word problems of Merrill et al. [71]. Each input is a sequence of elements of a finite group and the model must track their running composition. We use the symmetric groups S_3, S_4, and S_5 and the alternating group A_5. Following the DeltaProduct protocol, models are trained at length 128 and evaluated at length 512, so the task also measures length extrapolation. We match model depth, hidden size, recurrent-state size, and optimization across architectures and report the mean and standard deviation over three seeds.

DeltaProduct reports that the standard nonnegative transition range fails on these tasks. Gated DeltaNet, KDA, and Gated DeltaNet-2 therefore use their state-tracking-enabled gate range, $\beta_t \in [0, 2]$ for the scalar-gated models and $\boldsymbol{b}_t \in [0, 2]^{d_k}$ for Gated DeltaNet-2 as described in Section 3.1,

while Mamba-3 uses its data-dependent rotations [13]. This expanded erase range is already supported by the Gated DeltaNet-2 kernels, and Table 6 shows that it is neutral for language modeling and retrieval at 1.3B. We additionally report Gated DeltaNet-2 with the standard range $\boldsymbol{b}_t \in [0,1]^{d_k}$ as a control.

Model	$S_3 \uparrow$	$S_4 \uparrow$	$A_5 \uparrow$	$S_5 \uparrow$
Chance	16.7	4.2	1.7	0.8
Mamba-3	29.1 ± 1.7	7.2 ± 0.5	2.4 ± 0.2	1.0 ± 0.1
Gated DeltaNet, $\beta_t \in [0,2]$	45.0 ± 2.6	12.0 ± 0.8	3.8 ± 0.3	1.3 ± 0.1
KDA, $\beta_t \in [0,2]$	52.4 ± 2.2	15.8 ± 0.9	5.0 ± 0.4	1.5 ± 0.1
Gated DeltaNet-2, $\boldsymbol{b}_t \in [0,1]^{d_k}$	18.3 ± 0.6	4.5 ± 0.2	1.8 ± 0.1	0.9 ± 0.1
Gated DeltaNet-2, $\boldsymbol{b}_t \in [0,2]^{d_k}$	$\mathbf{74.2 \pm 1.8}$	$\mathbf{28.6 \pm 1.3}$	$\mathbf{9.7 \pm 0.6}$	$\mathbf{2.8 \pm 0.2}$

Table S.4: Accuracy on the DeltaProduct permutation-composition tasks, trained at length 128 and evaluated at length 512. Values are mean $\pm$ standard deviation over three seeds.

At length 512, Gated DeltaNet-2 with the expanded erase range is best on all four groups, improving over KDA by 21.8, 12.8, 4.7, and 1.3 points on S_3, S_4, A_5, and S_5, respectively. The standard-range variant remains near chance, which shows that the two mechanisms are complementary. Negative eigenvalues enable reflection-like state transitions, while decoupled erase and write gates make these transitions substantially easier to learn. The low absolute accuracy on S_5 also identifies higher-order permutation composition as a remaining limitation of all evaluated models.

J Decoding latency relative to KDA

Fig. 2 reports training throughput. Inference behavior is equally important for recurrent models, so this appendix quantifies the decoding cost of the two additional gates relative to KDA. Both models maintain the same fixed-size recurrent state, so decoding memory and per-token complexity are independent of the context length for both. The question is the constant-factor cost of forming $\boldsymbol{b}_t \odot \boldsymbol{k}_t$ and $\mathbf{w}_t \odot \boldsymbol{v}_t$ and of the asymmetric erase in the recurrent decoding kernel of Appendix C.5.

We report matched kernel-level decoding latency estimates on an NVIDIA Blackwell SM100 GPU with $H = 128$ heads and $d_k = d_v = 64$. All measurements use an 8K-token prefix and report the latency of one recurrent decoding step for batch sizes 1, 2, 8, and 16.

Batch	Gated DeltaNet-2 (ms/token) ↓	KDA (ms/token) ↓	Latency ratio ↓	Throughput ratio ↑
1	0.048	0.046	1.04×	0.96×
2	0.049	0.047	1.04×	0.96×
8	0.056	0.053	1.06×	0.95×
16	0.066	0.062	1.06×	0.94×

Table S.5: Modeled decoding latency of one recurrent step on an NVIDIA Blackwell SM100 GPU with an 8K-token prefix. The latency ratio is the Gated DeltaNet-2 latency divided by the KDA latency, and the throughput ratio is its inverse.

Table S.5 shows that Gated DeltaNet-2 incurs only a 4.3 to 6.5% modeled decoding-latency overhead and retains 93.9 to 95.9% of the KDA throughput across batch sizes. Thus the independent erase and write gates add only a modest constant-factor cost while preserving the efficient recurrent decoding behavior of KDA.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately summarize contributions and claims are substantiated by the methodology and experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are noted alongside the experiments, including the throughput and decoding-latency overhead relative to KDA (Section 4 and Appendix J), the lack of gain from the expanded erase range at the 1.3B scale (Table 6), and the low absolute accuracy of all evaluated models on higher-order permutation composition (Appendix I).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Derivations of the fast-weight objective, chunkwise WY algorithm, and gate-aware backward pass are given in Section 3 and Appendices A and B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 4 and Appendix E specify the training recipe, model configurations, and evaluation suites, and Appendices C and D document the implementation and numerical details.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is released at `https://github.com/NVlabs/GatedDeltaNet-2`. Section 4 and Appendices C to E provide the training recipe and implementation details, and all data and benchmarks used are publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and Appendix E specify the optimizer, schedule, batch size, sequence length, model configurations, and evaluation benchmarks.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars are not reported for the main 1.3B tables because pretraining every architecture on 100B tokens is computationally prohibitive to repeat across multiple seeds. The matched comparison in Appendix G and the state-tracking experiments in Appendix I report the mean and standard deviation over three seeds.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Training throughput on a single H100 is reported in Figure 2, decoding latency on a Blackwell GPU in Appendix J, kernel hardware dispatch in Appendix C.4, and the training recipe and model scales in Section 4 and Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses publicly available data and standard benchmarks and conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: The paper proposes a general-purpose architecture with no direct societal impact beyond that of language models in general.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: No pretrained weights or datasets are released; only an architecture trained on public FineWeb-Edu.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All baselines, datasets, and benchmarks are cited to their original sources in Sections 1 and 4 and Appendix E.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released code (`https://github.com/NVlabs/GatedDeltaNet-2`) is documented and distributed under the NVIDIA Source Code License-NC. No datasets or pretrained models are released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects or studies requiring IRB review.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were not used in this work.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.