Fast Weight Attention for Continual Learning

FWA Authors

March 9, 2026*

Abstract

Recurrent fast-weight memories and selective state-space models compress a growing context into a bounded state, their writes can therefore be viewed as online continual-learning rules. We study these rules under a read-after-write autoregressive convention, where the prefix-aligned local pair is $(\mathbf{x}_t, \mathbf{y}_t) = (\phi(\mathbf{k}_{t-1}), \mathbf{v}_t)$. The same-step pair $(\phi(\mathbf{k}_t), \mathbf{v}_t)$ remains causal, but it optimizes a different fast-memory objective. We make this distinction explicit and derive first-order updates for regression and inner-product memories with consistent semantics: β_t is a dimensionless plasticity gain, λ_t is the actual shrinkage/ridge coefficient, and η_t is the induced step size. For regression, η_t is matched to the local smoothness of the ridge objective. For inner-product writes, it is an energy-normalized write gain. The resulting family consists of three regression rules, FALCON-1 with a scalar NLMS ridge step, FALCON-2 with per-column NLMS ridge steps, and FALCON-3 with a sliding-window mini-batch regression step, and three inner-product counterparts: FALCON-1A with a scalar energy-normalized write, FALCON-2A with per-column energy-normalized writes, and FALCON-3A with a sliding-window inner-product write. We evaluate the scalar and sliding inner-product variants in language modeling tasks and variable-length addition tasks. The aligned, normalized updates preserve competitive language-model quality and improve arithmetic-length extrapolation.

1 Introduction

Transformers (Vaswani et al., 2017) dominate modern language modeling because self-attention effectively captures global dependencies. Their main limitation is cost: standard attention scales quadratically, $\mathcal{O}(N^2)$, in sequence length N. For long contexts, both the attention matrix and the memory traffic of the key-value (KV) cache become major bottlenecks.

Beyond efficiency, long-context modeling is also a continual-learning problem: the model must bind new evidence online without catastrophic interference. Transformers externalize this fast memory as a growing KV cache, whereas SSMs and fast-weight models compress it into a bounded recurrent state. In such architectures, the state-update rule acts as a local learning rule, and its temporal alignment determines whether the fast memory is trained on information that was actually available at prediction time (Sun et al., 2024; Liu et al., 2024a; Behrouz et al., 2024; Wang et al., 2025).

Linear Attention (Katharopoulos et al., 2020), Fast Weight Programmers and Delta Networks (Schlag et al., 2021), RWKV (Peng et al., 2023), and Mamba (Gu and Dao, 2023) show that

*Revised: May 25, 2026.

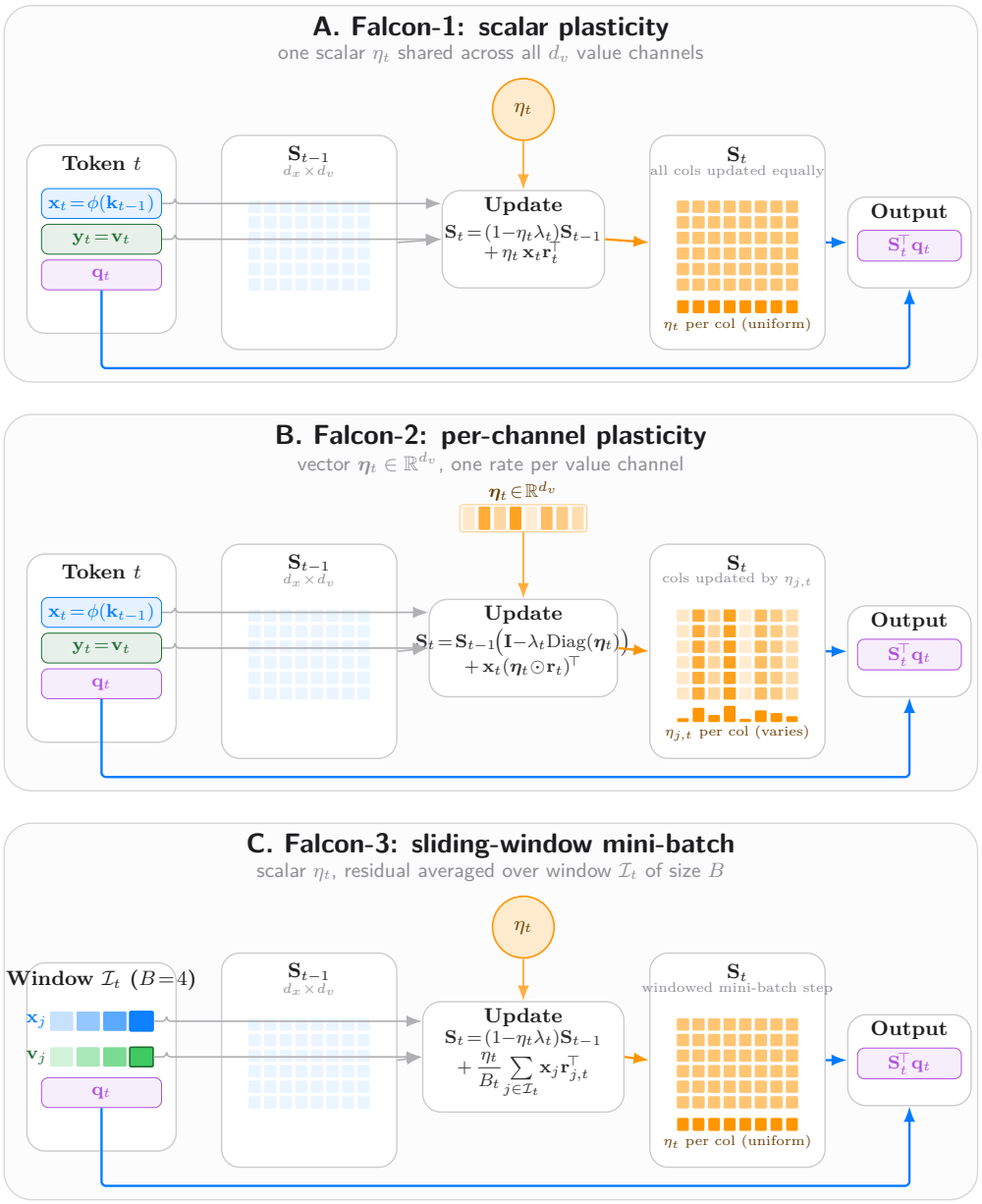

Figure 1 Falcon-1 vs. Falcon-2 vs. Falcon-3. Each panel shows one fast-weight update step on the matrix-valued state $\mathbf{S}_{t-1} \in \mathbb{R}^{d_x \times d_v}$, with residual $\mathbf{r}_t = \mathbf{v}_t - \mathbf{S}_{t-1}^\top \mathbf{x}_t$ (and $\mathbf{r}_{j,t} = \mathbf{v}_j - \mathbf{S}_{t-1}^\top \mathbf{x}_j$ inside the window). **(A) Falcon-1** applies one scalar η_t to all d_v value channels; every column of $\mathbf{S}_t$ receives the same plasticity (uniform orange shading and flat per-column bar). **(B) Falcon-2** promotes the step size to a vector $\boldsymbol{\eta}_t \in \mathbb{R}^{d_v}$ shown as the orange strip above the update card: each column of $\mathbf{S}_t$ is updated with its own gain $\eta_{j,t}$ (column shading and bar heights vary), while the write-feature $\mathbf{x}_t$ and residual $\mathbf{r}_t$ remain shared. **(C) Falcon-3** reverts to a scalar η_t but reads a sliding window $\mathcal{I}_t$ of B recent causal pairs; the blue/green strips show four $\{\mathbf{x}_j, \mathbf{v}_j\}_{j\in\mathcal{I}_t}$ entries (light $\rightarrow$ dark = older $\rightarrow$ newer; the boxed cell at $j=t$ is the current step). The update averages residuals over the window, giving a single windowed mini-batch step. The read $\mathbf{o}_t = \mathbf{S}_t^\top \mathbf{q}_t$ uses the updated state in all three cases.

such recurrent alternatives can be competitive while maintaining $\mathcal{O}(N)$ training and $\mathcal{O}(1)$ per-step inference. However, their write rules are typically specified heuristically.

Once read-after-write semantics are fixed, there is a convention mismatch: many recurrences bind the same-step pair $(\phi(\mathbf{k}_t), \mathbf{v}_t)$ (or $(\mathbf{k}_t, \mathbf{v}_t)$ when ϕ is the identity), whereas the prefix-prediction objective studied here pairs the newly revealed target with the prefix feature that was available when it was predicted, namely $(\phi(\mathbf{k}_{t-1}), \mathbf{v}_t)$, or equivalently $(\phi(\mathbf{k}_i), \mathbf{v}_{i+1})$ under standard indexing. The same-step pairing is still causal, but it optimizes a different internal fast-memory objective.

We therefore recast state-based sequence modeling as *autoregressive next-latent prediction*. The recurrent state $\mathbf{S}_t$ acts as a fast linear predictor from the prefix write feature $\mathbf{x}_t := \phi(\mathbf{k}_{t-1})$ to the newly revealed target $\mathbf{v}_t$. Building on Titans and ATLAS-style internal-memory views (Behrouz et al., 2024, 2025), we make this alignment explicit and derive normalized first-order updates that remain compatible with SSD-style chunk-parallel training (Dao and Gu, 2024). Our contributions are as follows:

- We identify the fast-memory training pair induced by read-after-write autoregressive modeling and separate it from the common same-step association.
- We derive FALCON-1, FALCON-2, and FALCON-3 as normalized first-order regression writes, with β_t controlling plasticity, λ_t controlling shrinkage, and η_t matched to the local smoothness scale.
- We derive inner-product counterparts, FALCON-1A, FALCON-2A, and FALCON-3A, using the same plasticity/forgetting semantics but interpreting the denominator as write-magnitude normalization rather than curvature normalization. Here FALCON-2A is the true per-column inner-product analogue of FALCON-2.
- We give chunk-parallel implementations for the resulting recurrences and evaluate the inner-product family in language modeling and arithmetic extrapolation.

2 Background

2.1 State Space Models

State-space representations have a long history in classical control, filtering, realization theory, and system identification. Twentieth-century foundations include Kalman filtering, state-variable realization, optimal filtering, and linear-systems theory (Kalman, 1960; Kalman and Bucy, 1961; Ho and Kálmán, 1966; Kung, 1978; Kung and Lin, 1981). Modern neural State Space Models (SSMs) (Gu and Dao, 2023; Dao and Gu, 2024) process a sequence of inputs $x(t) \in \mathbb{R}^{d_{\text{in}}}$ through a compressed latent state $\mathbf{h}(t) \in \mathbb{R}^n$, producing outputs $y(t) \in \mathbb{R}^{d_{\text{out}}}$. The continuous-time dynamics are linear in the hidden state once the system coefficients are fixed:

$$\dot{\mathbf{h}}(t) = \mathbf{A}(t; \mathbf{h}(t))\mathbf{h}(t) + \mathbf{B}(t; \mathbf{h}(t))x(t), \qquad y(t) = \mathbf{C}(t; \mathbf{h}(t))^\top \mathbf{h}(t),$$

where $\mathbf{A}(t; \mathbf{h}(t)) \in \mathbb{R}^{n \times n}$, $\mathbf{B}(t; \mathbf{h}(t)) \in \mathbb{R}^{n \times d_{\text{in}}}$, and $\mathbf{C}(t; \mathbf{h}(t)) \in \mathbb{R}^{n \times d_{\text{out}}}$. Equivalently, we write $\mathbf{A}(t)$, $\mathbf{B}(t)$, and $\mathbf{C}(t)$ as shorthand for coefficients that are data-dependent functions of the current state $\mathbf{h}(t)$.

Modern selective SSMs, such as Mamba (Gu and Dao, 2023), parametrize the system matrices $(\mathbf{B}, \mathbf{C}, \Delta)$ as functions of the current input x_t, allowing for content-aware filtering. Mamba-2 (Dao and Gu, 2024) further simplifies the transition matrix $\mathbf{A}$ to a scalar or diagonal structure, establishing a theoretical bridge known as Structured State Space Duality (SSD). This duality demonstrates

that the recurrent SSM is equivalent to a specific form of causal linear attention, enabling efficient training via chunk-parallel matrix multiplications while maintaining constant-state inference.

2.2 Linear Attention

Linear Attention (Katharopoulos et al., 2020) circumvents the $\mathcal{O}(N^2)$ complexity of standard attention by replacing the softmax with a kernel feature map $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^m$ such that $\kappa(\mathbf{q}_t, \mathbf{k}_j) = \phi(\mathbf{q}_t)^\top \phi(\mathbf{k}_j)$. Exploiting the associativity of matrix multiplication, the output $\mathbf{y}_t \in \mathbb{R}^{d_v}$ for the t-th token is:

$$\mathbf{y}_t = \frac{\sum_{j=1}^{t} \phi(\mathbf{q}_t)^\top \phi(\mathbf{k}_j)\mathbf{v}_j}{\sum_{j=1}^{t} \phi(\mathbf{q}_t)^\top \phi(\mathbf{k}_j)} = \frac{\left(\sum_{j=1}^{t} \phi(\mathbf{k}_j)\mathbf{v}_j^\top\right)^\top \phi(\mathbf{q}_t)}{\phi(\mathbf{q}_t)^\top \sum_{j=1}^{t} \phi(\mathbf{k}_j)}, \tag{2.1}$$

This formulation allows the context to be compressed into a recurrent matrix state $\mathbf{S}_t \in \mathbb{R}^{m \times d_v}$ and a normalizer $\mathbf{z}_t \in \mathbb{R}^m$:

$$\mathbf{y}_t = \frac{\mathbf{S}_t^\top \phi(\mathbf{q}_t)}{\mathbf{z}_t^\top \phi(\mathbf{q}_t) + \varepsilon_{\text{attn}}}, \qquad \mathbf{S}_t = \mathbf{S}_{t-1} + \phi(\mathbf{k}_t)\mathbf{v}_t^\top, \quad \mathbf{z}_t = \mathbf{z}_{t-1} + \phi(\mathbf{k}_t). \tag{2.2}$$

For a fresh sequence, Eq. (2.2) is exactly equivalent to Eq. (2.1) when $\mathbf{S}_0 = \mathbf{0}$, $\mathbf{z}_0 = \mathbf{0}$, $\varepsilon_{\text{attn}} = 0$, and the denominator is nonzero. With $\varepsilon_{\text{attn}} > 0$, it is the usual stabilized positive-feature variant. The normalized form is attention-like only when the read normalizer is nonnegative; signed-feature caveats are deferred to Appendix C.2.

Causality and indexing. Eq. (2.2) follows the standard Transformer convention: at position t we read from the updated state $(\mathbf{S}_t, \mathbf{z}_t)$, and the resulting representation is used to predict token $t{+}1$. Our next-latent alignment shifts the write stream by one: after observing $\mathbf{v}_t$, we write it under the previous write feature $\phi(\mathbf{k}_{t-1})$, or equivalently, $(\phi(\mathbf{k}_i), \mathbf{v}_{i+1})$ under standard indexing with $i = t - 1$. This yields the read-after-write recurrence

$$\mathbf{y}_t = \frac{\mathbf{S}_t^\top \phi(\mathbf{q}_t)}{\mathbf{z}_t^\top \phi(\mathbf{q}_t) + \varepsilon_{\text{attn}}}, \qquad \mathbf{S}_t = \mathbf{S}_{t-1} + \phi(\mathbf{k}_{t-1})\mathbf{v}_t^\top, \quad \mathbf{z}_t = \mathbf{z}_{t-1} + \phi(\mathbf{k}_{t-1}), \tag{2.3}$$

where $\varepsilon_{\text{attn}} \geq 0$ is a small stabilizer. Defining the shifted write-feature stream

$$\widetilde{\mathbf{x}}_1 := \mathbf{0}, \qquad \widetilde{\mathbf{x}}_t := \phi(\mathbf{k}_{t-1}) \quad \text{for } t \geq 2,$$

Eq. (2.3) is exactly Eq. (2.2) with the standard write-feature stream $\{\phi(\mathbf{k}_t)\}_{t=1}^T$ replaced by $\{\widetilde{\mathbf{x}}_t\}_{t=1}^T$ when $\varepsilon_{\text{attn}} = 0$; for $\varepsilon_{\text{attn}} > 0$, it is the corresponding stabilized shifted variant. The boundary condition is therefore imposed in feature space rather than raw-key space.

Normalized vs. unnormalized linear attention. The denominator in Eq. (2.1) uses the normalizer state $\mathbf{z}_t$ to rescale the readout. Many SSM/SSD-style architectures instead drop the denominator (and $\mathbf{z}_t$) and use an unnormalized inner-product read:

$$\mathbf{y}_t = \mathbf{S}_t^\top \phi(\mathbf{q}_t), \qquad \mathbf{S}_t = (1 - \eta_t \lambda_t)\mathbf{S}_{t-1} + \eta_t\, \phi(\mathbf{k}_{t-1})\mathbf{v}_t^\top, \tag{2.4}$$

with the unshifted convention recovered by replacing $\mathbf{k}_{t-1}$ with $\mathbf{k}_t$. In this denominator-free form, bounded state magnitude and a controllable memory timescale are enforced by explicit decay (e.g., $\lambda_t > 0$) and/or gain control. Section 4.3 shows that the numerator-state update in Eq. (2.4) is

exactly gradient descent on an inner-product objective. The auxiliary normalizer $\mathbf{z}_t$ in the normalized variants is a separate bookkeeping state for the read denominator; it is not itself obtained from that objective.

Accordingly, Eq. (2.4) is the gradient-descent update for the inner-product objective in Section 4.3. When $\lambda_t = 0$, the write is purely additive rank-one Hebbian learning; when $\lambda_t > 0$, it is additive plus scalar shrinkage. As noted in Section 2.1, Mamba-2 proves that the no-normalizer recurrence is functionally equivalent to a specific class of SSMs under the SSD framework.

2.3 Delta Networks

Fast Weight Programmers and Delta Networks (Schlag et al., 2021) formulate sequence modeling as the online learning of a value-retrieval function. Let $\mathbf{S}_{t-1} \in \mathbb{R}^{d \times d_v}$ denote the fast-weight state matrix. Instead of purely additive accumulation, standard Delta Networks employ an error-driven update derived from the gradient of the instantaneous squared error between the state's reconstruction of the current key and value:

$$\ell_t(\mathbf{S}) := \frac{1}{2} \left\| \mathbf{S}^\top \mathbf{k}_t - \mathbf{v}_t \right\|_2^2. \tag{2.5}$$

Here, $\mathbf{S}^\top \mathbf{k}_t$ represents the model's prediction of value $\mathbf{v}_t$ given key $\mathbf{k}_t$. The gradient with respect to the state is $\nabla_{\mathbf{S}} \ell_t(\mathbf{S}) = \mathbf{k}_t (\mathbf{S}^\top \mathbf{k}_t - \mathbf{v}_t)^\top$.

Delta rule. We denote the gradient step size by η_t; later, β_t denotes a dimensionless normalized gain and η_t the induced step size. A single online gradient step gives

$$\mathbf{S}_t = \mathbf{S}_{t-1} - \eta_t \, \mathbf{k}_t (\mathbf{S}_{t-1}^\top \mathbf{k}_t - \mathbf{v}_t)^\top = (\mathbf{I} - \eta_t \mathbf{k}_t \mathbf{k}_t^\top) \mathbf{S}_{t-1} + \eta_t \mathbf{k}_t \mathbf{v}_t^\top. \tag{2.6}$$

The rank-one factor performs a targeted shrinkage/edit along the current key direction; it becomes an orthogonal projection only when $\eta_t = 1/\|\mathbf{k}_t\|_2^2$. Gated Delta Networks add an explicit global decay gate; notation and comparison details are in Appendix C.3.

3 Autoregressive Next-Latent Prediction

In this section, we cast the recurrent write as an explicit online optimization problem. Under the read-after-write convention, the causal example revealed at step t pairs the newly observed target with the prefix write feature available when that target was predicted, namely $\mathbf{x}_t = \phi(\mathbf{k}_{t-1})$ and $\mathbf{y}_t = \mathbf{v}_t$. Standard DeltaNet instead uses the same-step pair $(\phi(\mathbf{k}_t), \mathbf{v}_t)$; that pairing remains causal, but it corresponds to a different local fast-memory objective. We therefore model $\mathbf{S}$ as an online linear predictor from $\mathbf{x}_t$ to $\mathbf{y}_t$ and optimize an instantaneous ridge-regression loss (Wang et al., 2025; Behrouz et al., 2024, 2025):

$$\ell_t(\mathbf{S}) \triangleq \frac{1}{2} \left\| \mathbf{S}^\top \mathbf{x}_t - \mathbf{y}_t \right\|_2^2 + \frac{\lambda_t}{2} \|\mathbf{S}\|_F^2, \tag{3.1}$$

where $\lambda_t \geq 0$ is a regularization coefficient. While full-batch minimization of the cumulative loss corresponds to the offline solution found in methods like MesaNet (von Oswald et al., 2025), efficient autoregressive modeling requires an online approximation. We therefore employ Online Gradient Descent (OGD).

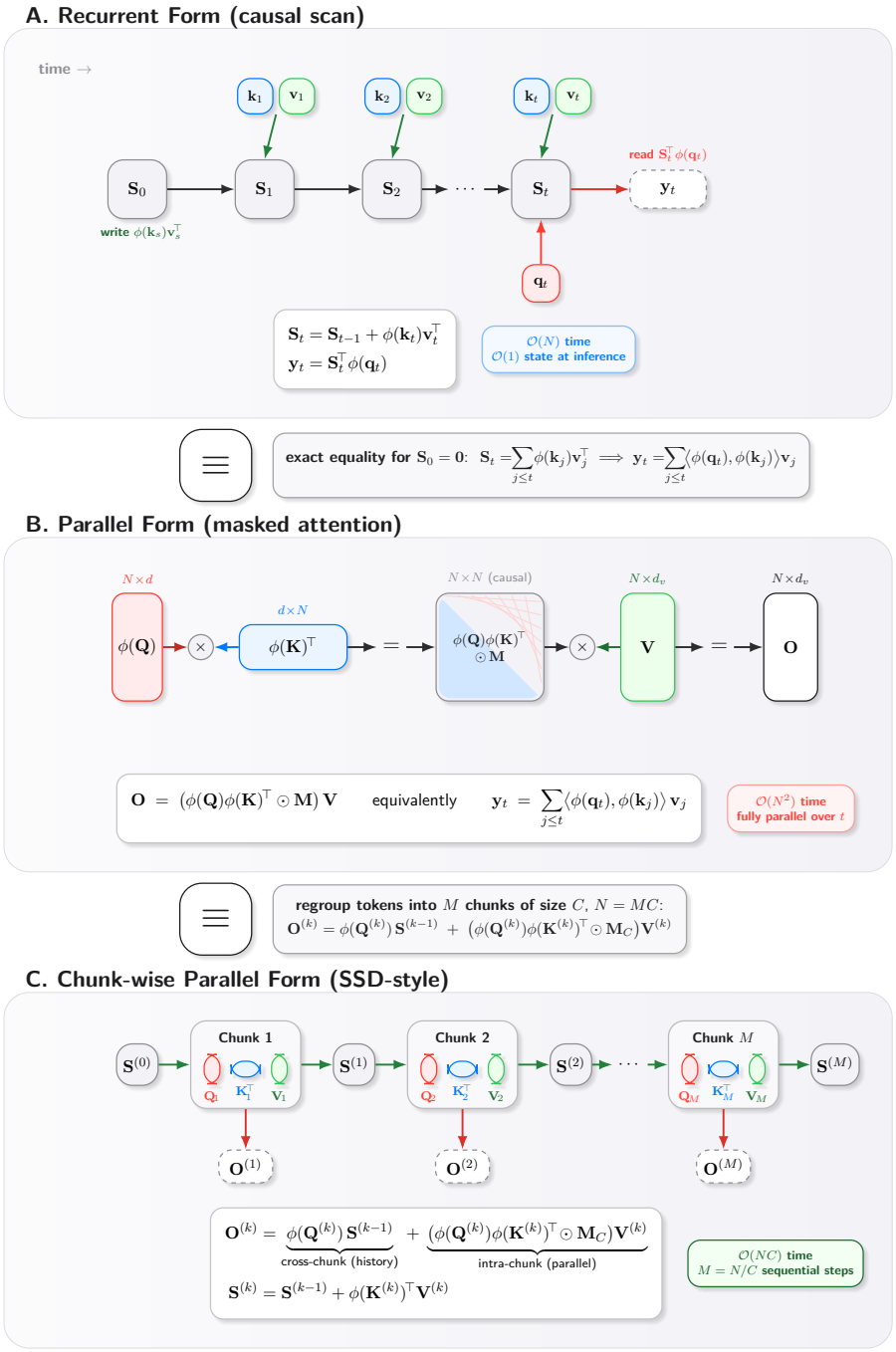

Figure 2 Three equivalent views of linear attention. (A) The recurrent form maintains a bounded matrix state $\mathbf{S}_t \in \mathbb{R}^{d\times d_v}$. (B) Unrolling the recurrence yields a masked-attention form with a causal mask $\mathbf{M}$: $\mathbf{O} = (\phi(\mathbf{Q})\phi(\mathbf{K})^\top \odot \mathbf{M})\mathbf{V}$. This costs $\mathcal{O}(N^2)$ time but is fully parallel over the sequence dimension. (C) The chunk-wise parallel form interpolates between (A) and (B): the sequence is split into M chunks of size C, intra-chunk computation is performed by parallel masked attention, and a compact state $\mathbf{S}^{(k)} \in \mathbb{R}^{d\times d_v}$ is propagated across chunks.

Notation. In kernelized linear attention, the key that writes to memory is typically a feature vector $\phi(\mathbf{k}) \in \mathbb{R}^m$ rather than the raw key $\mathbf{k} \in \mathbb{R}^d$. All derivations in this section are interpreted

$$\mathbf{x}_t \equiv \phi(\mathbf{k}_{t-1}) \in \mathbb{R}^m, \qquad \mathbf{S}_t \in \mathbb{R}^{m \times d_v},$$

so we treat $\mathbf{x}_t$ as the generic write feature and keep notation uncluttered. Queries used for retrieval, e.g., $\phi(\mathbf{q}_t)$ in linear attention, live in the same feature space but need not equal $\mathbf{x}_t$. Whenever a summary sentence informally refers to a pairing in raw-key space, the mathematically exact kernelized object is the corresponding write feature.

Implicit fast-memory objective. Eq. (3.1) is the instantaneous objective whose gradient step *defines* the fast-memory write rule; it is not an additional supervised loss beyond the outer autoregressive likelihood. During training, we differentiate through the update so that the slow weights (which produce $(\mathbf{q}, \mathbf{k}, \mathbf{v})$ as well as β_t, λ_t) learn representations and gates that make these local updates useful.

Indexing and causality conventions. We use the read-after-write (RAW) convention throughout: after token t is observed and written, the updated state $\mathbf{S}_t$ is read to predict token $t{+}1$. Under next-latent alignment, the causal write pair is therefore $(\phi(\mathbf{k}_{t-1}), \mathbf{v}_t)$, or equivalently $(\phi(\mathbf{k}_i), \mathbf{v}_{i+1})$ under standard indexing. We set $\mathbf{S}_0 = \mathbf{0}$ and impose the feature-space boundary $\mathbf{x}_1 := \mathbf{0}$. For update rules with explicit shrinkage, the boundary sentinel is also assigned $\eta_1 := 0$ (equivalently, $\alpha_1 = 0, \gamma_1 = 1$ in the log-space notation); otherwise $\lambda_1 > 0$ would decay a carried state even though no data pair is written. Detailed RAW/RBW and boundary conventions are deferred to Appendix C.5.

With this convention, the internal fast-memory prediction is $\widehat{\mathbf{y}}_t := \mathbf{S}_{t-1}^\top \mathbf{x}_t$, with residual $\mathbf{r}_t = \mathbf{y}_t - \widehat{\mathbf{y}}_t$. This is distinct from the model readout at position t, which under RAW uses the updated state $\mathbf{S}_t$.

Fast memory as continual learning. The recurrent state is the *fast* memory updated within the forward pass; each token provides a local training pair $(\mathbf{x}_t, \mathbf{y}_t)$. Across the family, β_t controls plasticity and λ_t controls shrinkage/forgetting; the realized η_t depends on the local normalization statistic, which is smoothness-matched for regression and energy-based for the inner-product implementations.

3.1 Online Gradient Descent Update

The gradient of the instantaneous loss in Eq. (3.1) with respect to the state $\mathbf{S}$ is:

$$\nabla_{\mathbf{S}} \ell_t(\mathbf{S}) = \mathbf{x}_t(\mathbf{S}^\top \mathbf{x}_t - \mathbf{y}_t)^\top + \lambda_t \mathbf{S}. \tag{3.2}$$

Applying a single gradient descent step with learning rate η_t yields the update rule:

$$\begin{aligned} \mathbf{S}_t &\leftarrow \mathbf{S}_{t-1} - \eta_t \nabla_{\mathbf{S}} \ell_t(\mathbf{S}_{t-1}) \\ &= \mathbf{S}_{t-1} - \eta_t \left[\mathbf{x}_t(\mathbf{S}_{t-1}^\top \mathbf{x}_t - \mathbf{y}_t)^\top + \lambda_t \mathbf{S}_{t-1} \right] \\ &= (1 - \eta_t \lambda_t)\mathbf{S}_{t-1} + \eta_t \mathbf{x}_t \mathbf{r}_t^\top, \end{aligned} \tag{3.3}$$

where $\mathbf{r}_t \triangleq \mathbf{y}_t - \mathbf{S}_{t-1}^\top \mathbf{x}_t$ is the residual (prediction error). Note that, unlike standard Delta Networks, $\mathbf{r}_t$ measures the discrepancy between the prediction from the previous write feature $\mathbf{x}_t = \phi(\mathbf{k}_{t-1})$ and the current value $\mathbf{v}_t$.

This is gradient descent on the instantaneous ridge objective. With the normalized step size below, it becomes a normalized update. In the special case $\lambda_t = 0$ and $\varepsilon = 0$, it reduces exactly to the classical NLMS recursion; for $\varepsilon > 0$, it is the usual stabilized NLMS variant. Specifically, the loss ℓ_t is L_t-smooth with respect to the Frobenius norm, with smoothness constant $L_t = \|\mathbf{x}_t\|_2^2 + \lambda_t$. To ensure numerical stability and scale robustness, we adopt the normalized step size:

$$\eta_t = \frac{\beta_t}{\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon}, \qquad \beta_t \in (0, 2),\ \varepsilon \geq 0. \tag{3.4}$$

We adopt the convention $\eta_t := 0$ when $\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon = 0$, and also at the boundary sentinel $t = 1$ when $\mathbf{x}_1 = \mathbf{0}$ is used to denote "no causal pair." In analysis, we may take $\varepsilon = 0$ and assume $L_t > 0$. In implementations, we take $\varepsilon > 0$ for numerical robustness; any $\varepsilon > 0$ only decreases η_t and therefore preserves the descent guarantees below.

Appendix C.6 records secondary implementation details, including the $\beta_t > 1$ sign-flip regime and the positive-decay interpretation of ridge shrinkage under log-space unrolling.

3.2 Analysis

We show that the normalized step size yields per-step descent in the instantaneous regularized objective ℓ_t, a basic local stability property. This statement is pointwise in t: it does not imply monotone decrease of the cumulative online loss $\sum_s \ell_s$ or of the outer autoregressive training objective.

Lemma 3.1 (Per-step Descent for Smooth Losses)**.** Let $f : \mathbb{R}^{d_x \times d_v} \to \mathbb{R}$ be L-smooth with respect to the Frobenius norm. For any step size $\eta \in (0, 2/L)$, the gradient step $\mathbf{S}^+ = \mathbf{S} - \eta \nabla f(\mathbf{S})$ satisfies

$$f(\mathbf{S}^+) \leq f(\mathbf{S}) - \frac{\eta(2 - \eta L)}{2} \|\nabla f(\mathbf{S})\|_F^2. \tag{3.5}$$

Proof. By L-smoothness, for any $\mathbf{S}$ and $\mathbf{S}'$,

$$f(\mathbf{S}') \leq f(\mathbf{S}) + \langle \nabla f(\mathbf{S}), \mathbf{S}' - \mathbf{S} \rangle + \frac{L}{2} \|\mathbf{S}' - \mathbf{S}\|_F^2.$$

Set $\mathbf{S}' = \mathbf{S}^+ = \mathbf{S} - \eta \nabla f(\mathbf{S})$ and simplify. □

Step-size parametrization. Choosing $\eta = \beta/L$ with $\beta \in (0, 2)$ (hence requiring $L > 0$) yields a decrease coefficient $\beta(2 - \beta)/(2L)$. For $L > 0$, the stabilized choice $\eta = \beta/(L + \varepsilon)$ with $\varepsilon \geq 0$ also lies in $(0, 2/L)$. When $L = 0$, this interval is undefined; in the degenerate cases arising here ($\mathbf{x}_t = \mathbf{0}$ and $\lambda_t = 0$, or the corresponding windowed analogue), the gradient is zero, so we define the update to be a no-op by setting $\eta := 0$.

3.3 Delta Networks as Regression

In this section, we interpret many previous *fast weight* models under this optimization framework. We observe that Delta Networks and Linear Attention are not merely architectural heuristics, but solutions to specific online objective functions.

By substituting the regression assignments $\mathbf{x}_t \leftarrow \phi(\mathbf{k}_{t-1})$ and $\mathbf{y}_t \leftarrow \mathbf{v}_t$, Eq. (3.3) recovers the functional form of the Delta Network update rule (Schlag et al., 2021), but with the critical index shift:

$$\mathbf{S}_t = \underbrace{\left((1-\eta_t\lambda_t)\mathbf{I}_{d_x} - \eta_t \mathbf{x}_t\mathbf{x}_t^\top\right)}_{\text{Decay \& Targeted Forget}} \mathbf{S}_{t-1} + \underbrace{\eta_t \mathbf{x}_t\mathbf{y}_t^\top}_{\text{Write}}, \qquad \mathbf{x}_t = \phi(\mathbf{k}_{t-1}),\ \mathbf{y}_t = \mathbf{v}_t. \tag{3.6}$$

Here, the rank-one term $\mathbf{x}_t\mathbf{x}_t^\top\mathbf{S}_{t-1}$ is the left Hessian action of the squared-error loss along the current write-feature direction. Intuitively, it reduces the component of the current predictor that acts on $\mathbf{x}_t = \phi(\mathbf{k}_{t-1})$ before adding the new target $\mathbf{y}_t = \mathbf{v}_t$.

In contrast, if we replace the regression (MSE) loss with an inner-product objective (Section 4.3), the residual term disappears, and the update becomes purely additive. With the standard (unshifted) assignment $(\mathbf{x}_t, \mathbf{y}_t) = (\phi(\mathbf{k}_t), \mathbf{v}_t)$ (or $(\mathbf{k}_t, \mathbf{v}_t)$ in the unkernelized case) this reduces to the familiar Linear Attention / Mamba-2 accumulation, with our next-latent assignment $(\mathbf{x}_t, \mathbf{y}_t) = (\phi(\mathbf{k}_{t-1}), \mathbf{v}_t)$ it yields the one-step-shifted variant used by our methods (e.g., Falcon-3A in Section 4.5).

This regression perspective motivates a key algorithmic improvement that we analyze in Section 4: with objective-matched normalization, the rank-1 regression step uses $L_t = \|\mathbf{x}_t\|_2^2 + \lambda_t$, while the sliding rule uses $L_t^{(B)} = \lambda_{\max}(\bar{\mathbf{C}}_t^{(B)}) + \lambda_t$. This suggests that fixed learning rates are scale-mismatched for regression-style fast-weight updates; for inner-product writes, the same normalization is better viewed as a magnitude stabilizer than as a curvature requirement.

Appendix A.2 gives reference pseudocode for the sequential first-order online-ridge update.

4 Falcon: Fast Weight Attention

Fast-weight memories and linear-attention architectures can be interpreted as online models that update a recurrent memory during the forward pass, tracing back to classical fast-weight mechanisms and their modern instantiations in linear Transformers and Delta-style rules (Hinton and Plaut, 1987; Schmidhuber, 1992; Ba et al., 2016; Schlag et al., 2021). In this section, we derive FALCON from two local objectives under the shifted $\phi(\mathbf{k}_{t-1}) \to \mathbf{v}_t$ alignment: squared-error regression and a negative inner-product objective. This yields normalized step sizes, explicit forgetting controls, and sliding-window variants.

Naming and formula summary. All variants use the same causal pair

$$\mathbf{x}_t := \phi(\mathbf{k}_{t-1}), \qquad \mathbf{y}_t := \mathbf{v}_t, \qquad \mathbf{x}_1 := \mathbf{0}, \qquad \eta_1 := 0,$$

and read after writing, $\mathbf{o}_t = \mathbf{S}_t^\top \phi(\mathbf{q}_t)$. Let $\mathbf{r}_t := \mathbf{y}_t - \mathbf{S}_{t-1}^\top \mathbf{x}_t$ and $\mathrm{Diag}(\boldsymbol{\eta}_t)$ denote the diagonal matrix of per-column step sizes. The regression family is

$$\text{FALCON-1:}\quad \mathbf{S}_t = (1-\eta_t\lambda_t)\mathbf{S}_{t-1} + \eta_t\,\mathbf{x}_t\mathbf{r}_t^\top, \qquad \eta_t = \frac{\beta_t}{\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon},$$

$$\text{FALCON-2:}\quad \mathbf{S}_t = \mathbf{S}_{t-1}(\mathbf{I}_{d_v} - \lambda_t\,\mathrm{Diag}(\boldsymbol{\eta}_t)) + \mathbf{x}_t(\boldsymbol{\eta}_t \odot \mathbf{r}_t)^\top, \qquad \eta_{j,t} = \frac{\beta_{j,t}}{\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon},$$

$$\text{FALCON-3:}\quad \mathbf{S}_t = (1-\eta_t\lambda_t)\mathbf{S}_{t-1} + \frac{\eta_t}{B_t}\sum_{j\in\mathcal{I}_t} \mathbf{x}_j(\mathbf{y}_j - \mathbf{S}_{t-1}^\top\mathbf{x}_j)^\top, \qquad \eta_t = \frac{\beta_t}{\mu_t^{(B)} + \lambda_t + \varepsilon}.$$

Here $\mu_t^{(B)} = \lambda_{\max}\Big(B_t^{-1}\sum_{j\in\mathcal{I}_t}\mathbf{x}_j\mathbf{x}_j^\top\Big)$. The inner-product family replaces the residual regression write by direct target writes:

$$\textsc{Falcon}\text{-1A:}\quad \mathbf{S}_t = (1-\eta_t\lambda_t)\mathbf{S}_{t-1} + \eta_t\,\mathbf{x}_t\mathbf{y}_t^\top, \qquad \eta_t = \frac{\beta_t}{E_t+\lambda_t+\varepsilon},$$

$$\textsc{Falcon}\text{-2A:}\quad \mathbf{S}_t = \mathbf{S}_{t-1}(\mathbf{I}_{d_v} - \lambda_t\,\mathrm{Diag}(\boldsymbol{\eta}_t)) + \mathbf{x}_t(\boldsymbol{\eta}_t\odot\mathbf{y}_t)^\top, \qquad \eta_{j,t} = \frac{\beta_{j,t}}{E_t+\lambda_t+\varepsilon},$$

$$\textsc{Falcon}\text{-3A:}\quad \mathbf{S}_t = (1-\eta_t\lambda_t)\mathbf{S}_{t-1} + \eta_t\,\bar{\mathbf{N}}_t^{(B)}, \qquad \eta_t = \frac{\beta_t}{\bar{E}_t^{(B)}+\lambda_t+\varepsilon},$$

where $E_t = \|\mathbf{x}_t\|_2^2$, $\bar{\mathbf{N}}_t^{(B)} = B_t^{-1}\sum_{j\in\mathcal{I}_t}\mathbf{x}_j\mathbf{y}_j^\top$, and $\bar{E}_t^{(B)} = B_t^{-1}\sum_{j\in\mathcal{I}_t}\|\mathbf{x}_j\|_2^2$. Thus, the index 1/2/3 denotes scalar, per-column, and sliding-window dynamics, respectively. The suffix "A" denotes the inner-product objective.

4.1 Scaled Linear Attention and Scaled DeltaNet

Unlike softmax attention, which uses the scaled dot product $\langle\mathbf{q},\mathbf{k}\rangle/\sqrt{d}$, fast-weight recurrences are directly sensitive to the norms of queries and keys: (i) dot-product reads grow with $\|\mathbf{q}_t\|_2\|\mathbf{k}\|_2$, and (ii) additive (inner-product) writes grow with the write-feature norm. To stabilize both the *read* and the *write* streams, especially under long decoding horizons and mixed precision, we use explicit feature scaling/normalization.

Scaled features. We define a generic RMS normalization operator for a vector $\mathbf{u}\in\mathbb{R}^{d_u}$:

$$\mathrm{RMSNorm}(\mathbf{u}) := \frac{\mathbf{u}}{\sqrt{\|\mathbf{u}\|_2^2/d_u + \varepsilon_{\mathrm{rms}}}},$$

where $\varepsilon_{\mathrm{rms}} > 0$ is a small stabilizer. Unless stated otherwise, we apply RMSNorm to the $(\mathbf{q}_t,\mathbf{k}_t)$ projections used by fast-weight reads/writes, before forming dot products or outer products. This default differs from common ℓ_2-normalized DeltaNet variants (e.g., in Gated DeltaNet implementations) and is substantially more stable in mixed precision because standard RMSNorm keeps coordinate magnitudes $\Theta(1)$. Moreover,

$$\|\,\mathrm{RMSNorm}(\mathbf{u})\|_2^2 = \frac{d_u\|\mathbf{u}\|_2^2}{\|\mathbf{u}\|_2^2 + d_u\varepsilon_{\mathrm{rms}}} \le d_u,$$

so in the usual regime $\|\mathbf{u}\|_2^2 \gg d_u\varepsilon_{\mathrm{rms}}$ we indeed have $\|\,\mathrm{RMSNorm}(\mathbf{u})\|_2^2 \approx d_u$. By default, we do *not* normalize values $\mathbf{v}_t$; value normalization (VNorm) is optional and disabled unless explicitly enabled.

Scaled Linear Attention. Under next-latent alignment, we use the RMS-normalized projections in the feature space. The scalar denominator-free inner-product recurrence, denoted Falcon-1A below, is

$$\mathbf{y}_t = \mathbf{S}_t^\top\phi(\mathbf{q}_t), \qquad \mathbf{S}_t = (1-\eta_t\lambda_t)\mathbf{S}_{t-1} + \eta_t\,\mathbf{x}_t\mathbf{v}_t^\top, \qquad \mathbf{x}_t := \phi(\mathbf{k}_{t-1}).$$

Its per-column counterpart, Falcon-2A, replaces the scalar write gain by a vector $\boldsymbol{\eta}_t\in\mathbb{R}^{d_v}$:

$$\mathbf{S}_t = \mathbf{S}_{t-1}(\mathbf{I}_{d_v} - \lambda_t\,\mathrm{Diag}(\boldsymbol{\eta}_t)) + \mathbf{x}_t(\boldsymbol{\eta}_t\odot\mathbf{v}_t)^\top.$$

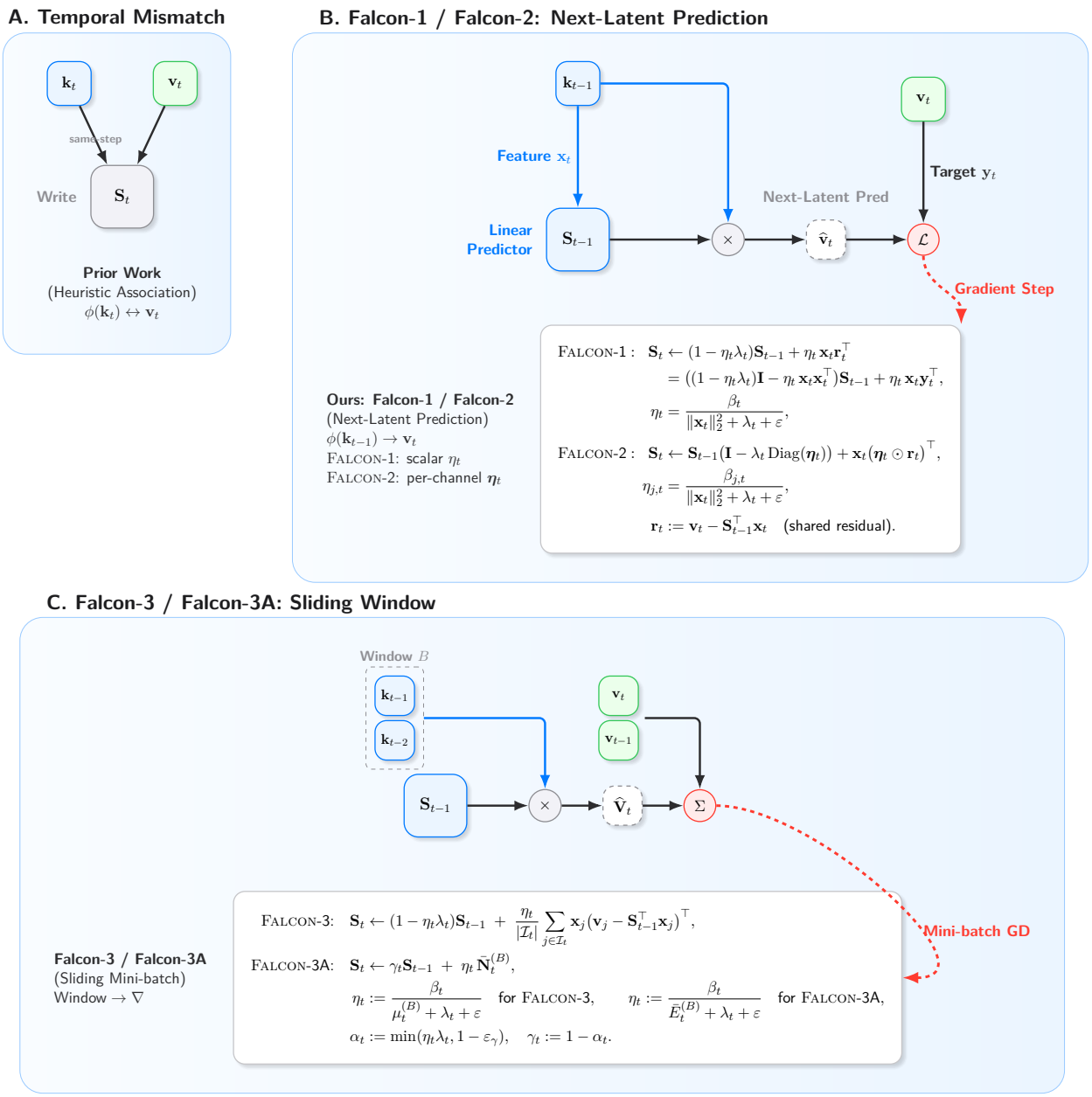

Figure 3 Conceptual Overview. (A) Many fast-weight rules bind the same-step write-feature/target pair $(\phi(\mathbf{k}_t), \mathbf{v}_t)$ as a cache-style association. Under the prefix-prediction fast-memory objective studied here, the example revealed at step t pairs a prefix write feature with the newly observed target, yielding $(\phi(\mathbf{k}_{t-1}), \mathbf{v}_t)$. (B) **Falcon-1 / Falcon-2 (Next-Latent Prediction):** the state $\mathbf{S}_{t-1}$ predicts $\mathbf{v}_t$ from the prefix feature $\mathbf{x}_t := \phi(\mathbf{k}_{t-1})$, then performs an online NLMS-stabilized ridge-regression update. FALCON-1 uses a single scalar step size η_t shared across all value channels (shared dynamics); FALCON-2 extends this to per-channel step sizes $\boldsymbol{\eta}_t \in \mathbb{R}^{d_v}$, allowing each value channel to follow its own plasticity trajectory while sharing the write-feature direction. Both share the residual $\mathbf{r}_t := \mathbf{v}_t - \mathbf{S}_{t-1}^\top \mathbf{x}_t$. (C) **Falcon-3 / Falcon-3A (Sliding Mini-batch):** applies a mini-batch regression (or inner-product) step over an active window $\mathcal{I}_t$ of nominal size B (realized size $B_t := |\mathcal{I}_t| \le B$) over causal pairs $(\mathbf{x}_j, \mathbf{v}_j)$ with $\mathbf{x}_j := \phi(\mathbf{k}_{j-1})$.

The corresponding normalized numerator/denominator recurrence is recorded in Appendix C.1; in signed-feature settings, the denominator caveat in Appendix C.2 applies.

Scaled DeltaNet and the common ridge parameterization. For regression-style fast weights, we use the same scaled write-feature $\mathbf{x}_t$ inside the NLMS update:

$$\mathbf{S}_t = (1 - \eta_t \lambda_t)\mathbf{S}_{t-1} + \eta_t\, \mathbf{x}_t(\mathbf{v}_t - \mathbf{S}_{t-1}^\top \mathbf{x}_t)^\top, \qquad \eta_t = \frac{\beta_t}{\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon}.$$

In scaled implementations, the network may emit a dimensionless base ridge $\bar{\lambda}_t$ that is converted to the actual coefficient used by the recurrence, $\lambda_t = \bar{\lambda}_t E_t$, where E_t is the appropriate normalization statistic: $\|\mathbf{x}_t\|_2^2$ for FALCON-2 and $\mu_t^{(B)} := \lambda_{\max}(\bar{\mathbf{C}}_t^{(B)})$ for FALCON-3. We then set

$$\eta_t = \frac{\beta_t}{E_t + \lambda_t + \varepsilon}, \qquad \alpha_t := \eta_t \lambda_t, \qquad \gamma_t := 1 - \alpha_t.$$

Here $E_t = \|\mathbf{x}_t\|_2^2$ for the non-sliding scalar/per-column rules (FALCON-1/FALCON-2 and FALCON-1A/FALCON-2A), while $E_t = \mu_t^{(B)}$ for FALCON-3 and $E_t = \bar{E}_t^{(B)}$ for FALCON-3A. For regression, E_t is the local smoothness scale of the data term; for inner-product writes, the corresponding energy statistics in Sections 4.3 and 4.5 are practical write-magnitude controls. Exact scale-robustness identities, detached-statistics details, the normalized linear-attention recurrence, and log-space positive-decay handling are deferred to Appendix C.1 and Appendix C.6.

4.2 Regression Loss (Delta Network)

We formulate the state update as an autoregressive linear regression problem. At time step t, let the state be $\mathbf{S}_{t-1} \in \mathbb{R}^{d_x \times d_v}$, the write feature be $\mathbf{x}_t \triangleq \phi(\mathbf{k}_{t-1}) \in \mathbb{R}^{d_x}$, and the target be $\mathbf{y}_t \triangleq \mathbf{v}_t \in \mathbb{R}^{d_v}$. (In the unkernelized case, ϕ is the identity and $d_x = d$.) The state $\mathbf{S}_{t-1}$ acts as a linear predictor mapping the prefix feature $\mathbf{x}_t$ to the newly observed target $\mathbf{y}_t$.

The instantaneous squared-error loss is

$$f_t(\mathbf{S}) := \frac{1}{2}\|\mathbf{S}^\top \mathbf{x}_t - \mathbf{y}_t\|_2^2. \tag{4.1}$$

Evaluated at the pre-update state, the residual is $\mathbf{r}_t \triangleq \mathbf{y}_t - \mathbf{S}_{t-1}^\top \mathbf{x}_t$. The gradient with respect to the state is:

$$\nabla_{\mathbf{S}} f_t(\mathbf{S}_{t-1}) = \mathbf{x}_t(\mathbf{S}_{t-1}^\top \mathbf{x}_t - \mathbf{y}_t)^\top = -\,\mathbf{x}_t \mathbf{r}_t^\top. \tag{4.2}$$

Delta update (standard vs. next-latent). A single step of Online Gradient Descent (OGD) with learning rate η_t yields the Delta update:

$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_{t-1} - \eta_t \nabla_{\mathbf{S}} f_t(\mathbf{S}_{t-1}) \\ &= \mathbf{S}_{t-1} + \eta_t \mathbf{x}_t \mathbf{r}_t^\top \\ &= (\mathbf{I}_{d_x} - \eta_t \mathbf{x}_t \mathbf{x}_t^\top)\mathbf{S}_{t-1} + \eta_t \mathbf{x}_t \mathbf{y}_t^\top. \end{aligned} \tag{4.3}$$

Under next-latent alignment we have $\mathbf{x}_t = \phi(\mathbf{k}_{t-1})$ and $\mathbf{y}_t = \mathbf{v}_t$. Replacing $\mathbf{k}_{t-1}$ with $\mathbf{k}_t$ (equivalently, $\mathbf{x}_t \leftarrow \phi(\mathbf{k}_t)$) recovers the unshifted DeltaNet update of Schlag et al. (2021).

L_2 Regularization. Adding an L_2 penalty $\frac{\lambda_t}{2}\|\mathbf{S}\|_F^2$ to the instantaneous loss yields the shrinkage term $-\eta_t \lambda_t \mathbf{S}_{t-1}$ in the online update (cf. Eq. (3.3)). Under normalized step sizes, this results in the multiplicative factor $(1 - \eta_t \lambda_t)$, which we treat as a simple and controllable forgetting mechanism.

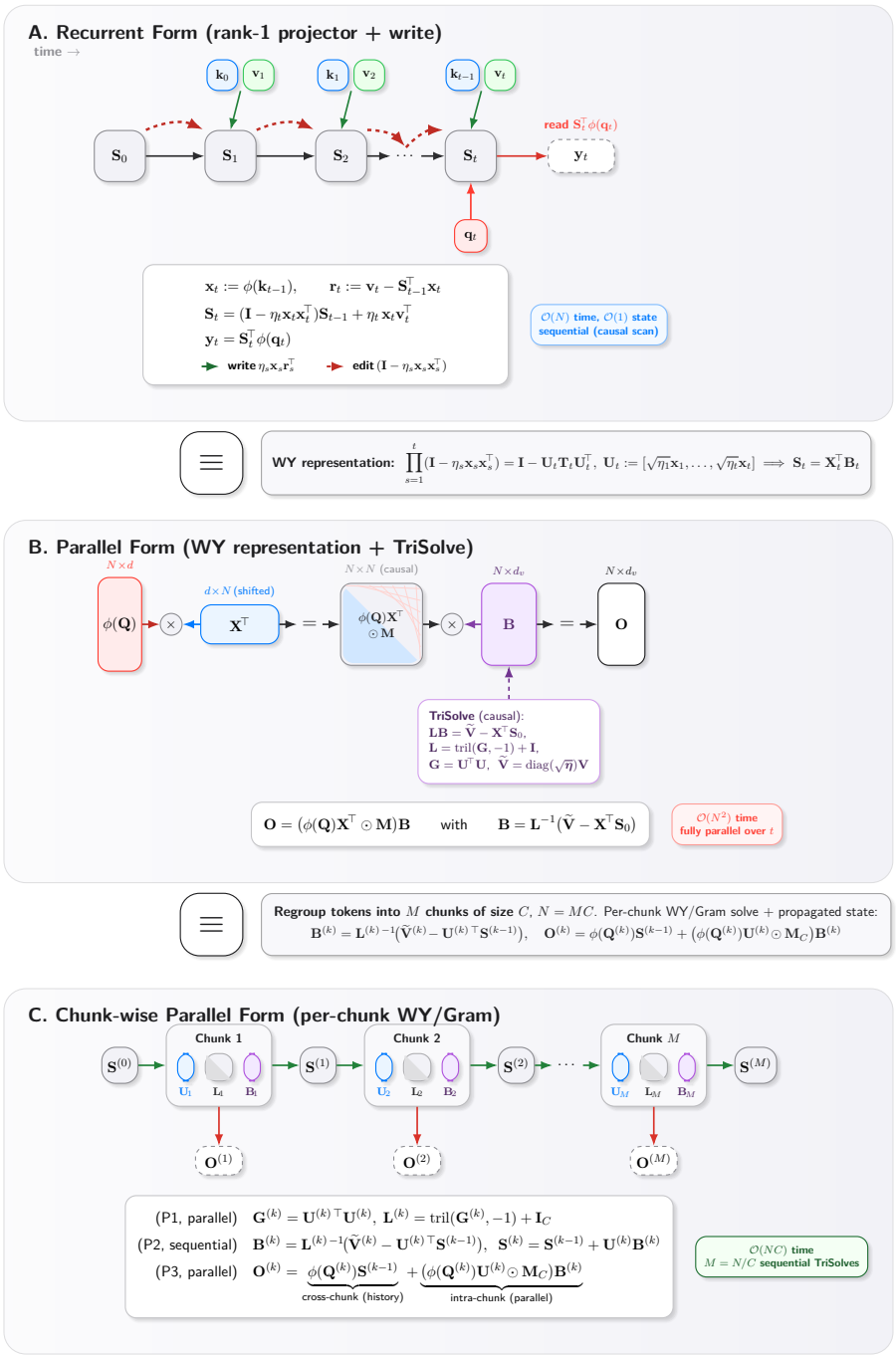

Figure 4 Three equivalent views of Falcon-1. (A) The recurrent form maintains a bounded matrix state $\mathbf{S}_t \in \mathbb{R}^{d_x \times d_v}$ that is simultaneously edited by a rank-1 projector $(\mathbf{I} - \eta_t \mathbf{x}_t \mathbf{x}_t^\top)$ along the current write-feature direction and rewritten with the new target $\mathbf{v}_t$ via the shifted pair $\mathbf{x}_t := \phi(\mathbf{k}_{t-1})$. (B) Unrolling the rank-1 projector product into the WY form $\prod_s (\mathbf{I} - \eta_s \mathbf{x}_s \mathbf{x}_s^\top) = \mathbf{I} - \mathbf{U}\mathbf{T}\mathbf{U}^\top$ converts the recurrence into a masked-attention pattern $\big(\phi(\mathbf{Q})\mathbf{X}^\top \odot \mathbf{M}\big)\mathbf{B}$. (C) The chunk-wise form interpolates between (A) and (B): the sequence is split into M chunks of size C, each chunk's WY/Gram system is built and solved in parallel.

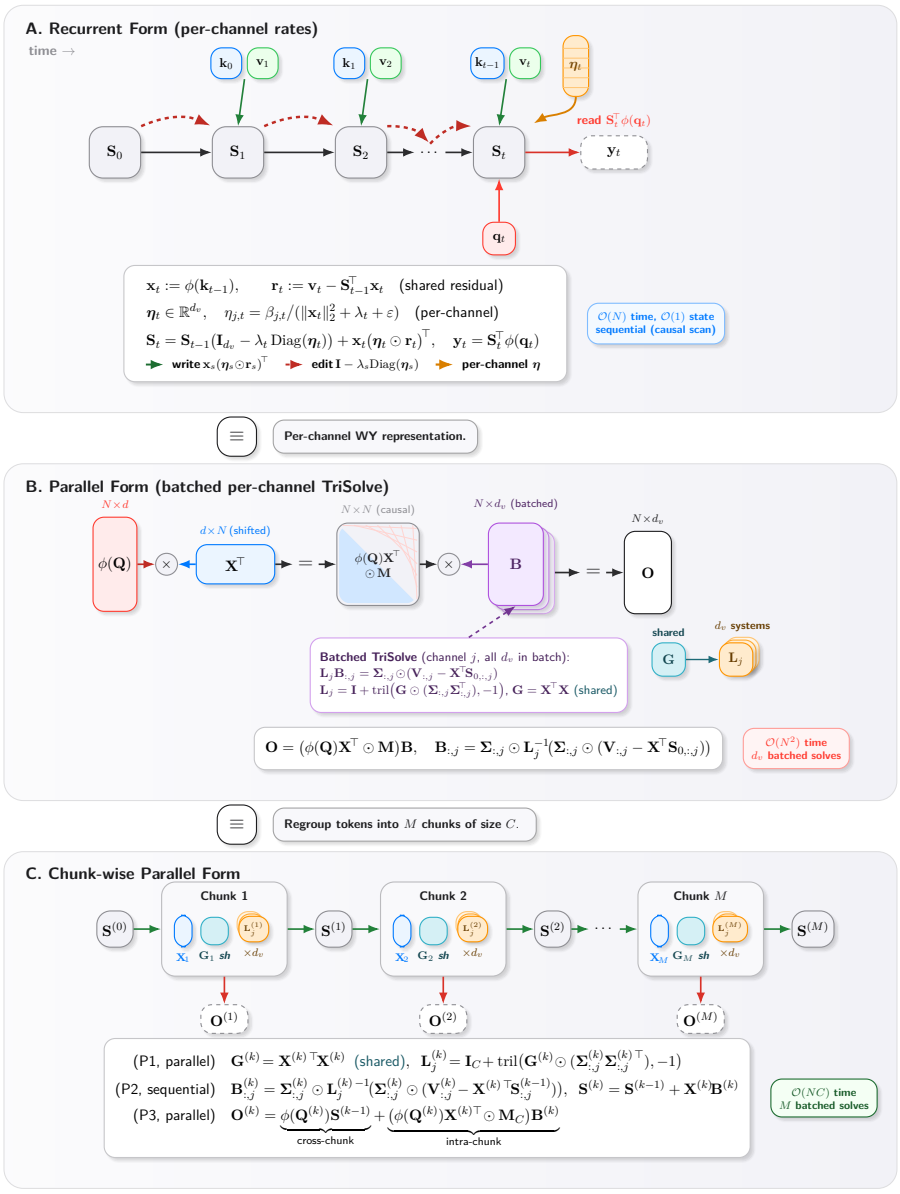

Figure 5 Three equivalent views of Falcon-2. (A) The recurrent form maintains a bounded matrix state $\mathbf{S}_t \in \mathbb{R}^{d_x \times d_v}$. Unlike Falcon-1, the step size is a vector $\boldsymbol{\eta}_t \in \mathbb{R}^{d_v}$, so each value channel follows its own plasticity trajectory while sharing the write-feature direction $\mathbf{x}_t = \phi(\mathbf{k}_{t-1})$ and the residual $\mathbf{r}_t = \mathbf{v}_t - \mathbf{S}_{t-1}^\top \mathbf{x}_t$. (B) The dual WY/Gram representation factors the recurrence as $\mathbf{S}_t = \mathbf{X}_t^\top \mathbf{B}_t$. Because $\boldsymbol{\eta}$ is per-channel, $\mathbf{B}$ is computed by a batched triangular solve with one unit-lower-triangular system $\mathbf{L}_j = \mathbf{I} + \mathrm{tril}(\mathbf{G} \odot (\boldsymbol{\Sigma}_{:,j} \boldsymbol{\Sigma}_{:,j}^\top), -1)$ per value channel, all sharing the same write-feature Gram $\mathbf{G} = \mathbf{X}^\top \mathbf{X}$. (C) The chunk-wise form splits the sequence into M chunks of size C. Within each chunk, the Gram $\mathbf{G}^{(k)}$ is built once and shared across channels; the d_v channel-specific systems $\mathbf{L}_j^{(k)}$ are then solved in a single batched TriSolve. A compact matrix state $\mathbf{S}^{(k)} \in \mathbb{R}^{d_x \times d_v}$ is propagated across chunks.

Falcon-2: Adaptive Learning Rates. We propose **Falcon-2**, which combines NLMS-style normalization with per-channel adaptive learning rates. In Appendix F, we derive a vectorized dual

form that makes column-wise adaptivity computationally tractable on GPUs. This allows the step size to be a vector $\boldsymbol{\eta}_t \in \mathbb{R}^{d_v}$, tailoring the update magnitude for each value channel (column of $\mathbf{S}$) independently. In the models studied here, we use the standard multi-head factorization: each head carries its own fast state, and the same per-column rule is applied independently within that head.

Per-column NLMS step sizes. Concretely, we use an NLMS-style normalizer shared across columns and learn per-column gains (equivalently, per output-feature / column gains):

$$\eta_{j,t} = \frac{\beta_{j,t}}{\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon}, \qquad \beta_{j,t} \in (0,2),\ \varepsilon > 0.$$

When $\lambda_t = 0$, $\varepsilon = 0$, and the gains are tied across channels ($\beta_{j,t} \equiv \beta_t$), this reduces to the classical scalar NLMS step size; otherwise, it is a column-wise NLMS generalization with a shared normalizer. Since the squared-error (ridge) loss decomposes across value coordinates (columns of $\mathbf{S}$), the per-step descent argument from Lemma 3.1 applies column-wise provided $0 < \beta_{j,t} < 2$ for all j. Equivalently, Falcon-2 is a collection of d_v independent scalar-step updates on a separable objective, rather than a single Frobenius-gradient step with a full matrix-valued learning rate.

Update rule. Let $\boldsymbol{\eta}_t = (\eta_{1,t}, \ldots, \eta_{d_v,t})^\top$. The update can be written compactly as:

$$\mathbf{S}_t = \mathbf{S}_{t-1}\Big(\mathbf{I}_{d_v} - \lambda_t\ \mathrm{Diag}(\boldsymbol{\eta}_t)\Big) + \mathbf{x}_t(\boldsymbol{\eta}_t \odot \mathbf{r}_t)^\top, \tag{4.4}$$

An expanded edit decomposition, separating the per-column shrinkage path from the feature-direction edit, is deferred to Appendix F.

Relation to RWKV-7 and Kimi Linear Attention. A brief comparison to RWKV-7 and Kimi Linear Attention is moved to Appendix C.6.

The equivalent column-wise recursion and the log-space positive-decay renormalization used by the chunk-parallel kernels are deferred to Appendix F.

Chunk-parallel implementation. Falcon-2 can be trained sequence-parallel by chunking the length axis and using a Gram/WY representation within each chunk. In the multi-head setting used here, this computation is applied independently within each head. Within a chunk, the WY/Gram form builds a shared key Gram matrix and a channel-dependent unit-lower-triangular system. The injected-value and projected-history paths share this triangular factor, so the implementation solves one merged residual system rather than two. This removes one batched TriSolve per chunk in the forward pass without changing the recurrence or asymptotic complexity, and mirrors the single-inversion form emphasized in Comba (Hu et al., 2025). Algorithm 1 gives the single-head Falcon-2 chunk-wise forward pass with the same positive-decay convention used by the implementation. The projector-only case is recovered by setting $\lambda_t = 0$, in which case $\gamma_{t,j} = 1$ and the chunk-local rescaling becomes the identity. Appendix F gives the exact WY/Gram algebra and complexity analysis.

Algorithm 1 FALCON-2 (Chunk-parallel Forward)

Require: Shifted write features $\mathbf{K} \in \mathbb{R}^{L \times d_x}$ with row t equal to $\mathbf{x}_t^\top$, query features $\mathbf{Q} \in \mathbb{R}^{L \times d_x}$, values $\mathbf{V} \in \mathbb{R}^{L \times d_v}$, per-channel step sizes $\boldsymbol{\eta} \in \mathbb{R}^{L \times d_v}$ with $\eta_{t,j} \geq 0$ (and $\eta_{1,:} = 0$ for the boundary sentinel), ridge coefficients $\boldsymbol{\lambda} \in \mathbb{R}^L_{\geq 0}$, decay floor $\varepsilon_\gamma > 0$, chunk size C (assume $C \mid L$), and initial state $\mathbf{S}_{\text{init}} \in \mathbb{R}^{d_x \times d_v}$.

Ensure: Outputs $\mathbf{O} \in \mathbb{R}^{L \times d_v}$ for the per-channel recurrence $\mathbf{s}_{t,j} = ((1-\alpha_{t,j})\mathbf{I} - \eta_{t,j}\mathbf{x}_t\mathbf{x}_t^\top)\mathbf{s}_{t-1,j} + \eta_{t,j}v_{t,j}\mathbf{x}_t$, where $\alpha_{t,j} := \min(\lambda_t\eta_{t,j}, 1-\varepsilon_\gamma)$; if the clamp is inactive, this is the ridge update with coefficient λ_t, and otherwise it uses the effective shrinkage coefficient $\alpha_{t,j}/\eta_{t,j}$ when $\eta_{t,j} > 0$.

Partition the length-L axis into $M = L/C$ contiguous chunks.
$\mathbf{S}_{\text{in}} \leftarrow \mathbf{S}_{\text{init}}$ and initialize $\mathbf{O} \leftarrow \mathbf{0}$.
for $m = 1, \dots, M$ **do**
  Slice $\mathbf{K}_{(m)}, \mathbf{Q}_{(m)} \in \mathbb{R}^{C \times d_x}$, $\mathbf{V}_{(m)} \in \mathbb{R}^{C \times d_v}$, $\boldsymbol{\eta}_{(m)} \in \mathbb{R}^{C \times d_v}$, and $\boldsymbol{\lambda}_{(m)} \in \mathbb{R}^C$.
  Set $\mathbf{K} \leftarrow \mathbf{K}_{(m)}^\top \in \mathbb{R}^{d_x \times C}$ and $\mathbf{Q} \leftarrow \mathbf{Q}_{(m)}^\top \in \mathbb{R}^{d_x \times C}$.
  $\boldsymbol{\alpha} \leftarrow \min\big(\boldsymbol{\eta}_{(m)} \odot (\boldsymbol{\lambda}_{(m)}\mathbf{1}_{d_v}^\top),\, 1-\varepsilon_\gamma\big)$.
  $\log\boldsymbol{\gamma} \leftarrow \mathrm{log1p}(-\boldsymbol{\alpha})$ and $\widehat{\boldsymbol{\eta}} \leftarrow \boldsymbol{\eta}_{(m)} \odot \exp(-\log\boldsymbol{\gamma})$.
  Compute chunk-local log-prefixes $u_{0,:} \leftarrow \mathbf{0}$ and, for $i = 1{:}C$, $u_{i,:} \leftarrow u_{i-1,:} + \log\boldsymbol{\gamma}_{i,:}$ and $\boldsymbol{\delta}_{i,:} \leftarrow \exp(u_{i,:})$.
  Rescale values locally: $\widehat{\mathbf{V}}_{i,:} \leftarrow \mathbf{V}_{(m),i,:} \odot \exp(-u_{i-1,:})$ for $i = 1{:}C$.
  $\boldsymbol{\Sigma} \leftarrow \sqrt{\widehat{\boldsymbol{\eta}}}$, $\mathbf{G} \leftarrow \mathbf{K}^\top\mathbf{K}$, and $\mathbf{M} \leftarrow \mathrm{tril}(\mathbf{Q}^\top\mathbf{K}, 0)$.
  $\mathbf{H} \leftarrow \mathbf{Q}^\top\mathbf{S}_{\text{in}}$ and $\mathbf{P} \leftarrow \mathbf{K}^\top\mathbf{S}_{\text{in}}$.
  For each value channel j, form $\mathbf{L}_j \leftarrow \mathbf{I}_C + \mathrm{tril}\big(\mathbf{G} \odot (\boldsymbol{\Sigma}_{:,j}\boldsymbol{\Sigma}_{:,j}^\top), -1\big)$.
  $\widetilde{\mathbf{B}} \leftarrow \mathrm{BatchedSolveTri}\big(\{\mathbf{L}_j\}_{j=1}^{d_v},\, \boldsymbol{\Sigma} \odot (\widehat{\mathbf{V}} - \mathbf{P})\big)$.
  $\mathbf{B} \leftarrow \boldsymbol{\Sigma} \odot \widetilde{\mathbf{B}}$.
  $\widehat{\mathbf{O}}^{(m)} \leftarrow \mathbf{H} + \mathbf{M}\mathbf{B}$ and $\widehat{\mathbf{S}}_{\text{out}} \leftarrow \mathbf{S}_{\text{in}} + \mathbf{K}\mathbf{B}$.
  Write $\mathbf{O}_{(m),i,:} \leftarrow \widehat{\mathbf{O}}_{i,:}^{(m)} \odot \boldsymbol{\delta}_{i,:}$ for $i = 1{:}C$.
  $\mathbf{S}_{\text{in}} \leftarrow \widehat{\mathbf{S}}_{\text{out}}\,\mathrm{Diag}(\boldsymbol{\delta}_{C,:})$.
end for
return $\mathbf{O}$.

4.3 Inner Product Loss (Linear Attention and Mamba-2)

We now consider an Inner Product objective that encourages alignment between the state prediction and the target. We write it in minimization form and optionally add an L_2 penalty:

$$\ell_t^{\text{ip}}(\mathbf{S}) \triangleq -\langle \mathbf{S}^\top\mathbf{x}_t, \mathbf{y}_t\rangle + \frac{\lambda_t}{2}\|\mathbf{S}\|_F^2, \qquad \lambda_t \geq 0. \tag{4.5}$$

When $\lambda_t = 0$, the objective is linear in $\mathbf{S}$ (no finite minimizer) and gradient descent reduces to purely additive Hebbian writes.

Standard vs. next-latent alignment. If we choose the unshifted features $(\mathbf{x}_t, \mathbf{y}_t) = (\phi(\mathbf{k}_t), \mathbf{v}_t)$ (or $(\mathbf{k}_t, \mathbf{v}_t)$ in the unkernelized case), the additive update below matches the usual Linear Attention write $\phi(\mathbf{k}_t)\mathbf{v}_t^\top$ (Eq. (2.2)). Our next-latent framework instead uses $(\mathbf{x}_t, \mathbf{y}_t) = (\phi(\mathbf{k}_{t-1}), \mathbf{v}_t)$, yielding a one-step shifted write stream.

Falcon-A variants and notation. We use the suffix "A" for the inner-product objective and keep the numerical index aligned with the regression family. Thus Falcon-1A is the scalar non-sliding inner-product rule, Falcon-2A is the true per-column non-sliding inner-product rule, and Falcon-3A is the sliding-window inner-product rule in Section 4.5. As in Falcon-1/Falcon-2/Falcon-3, β_t denotes the dimensionless gain, λ_t the actual shrinkage coefficient used by the recurrence (obtained directly or via the same scale-coupled construction described above), and η_t the resulting step size. For the inner-product family, this step size should be read as an energy-normalized write gain rather than as a curvature-matched denominator. Any decay fraction $\alpha_t := \eta_t\lambda_t$ is derived rather than independently parameterized.

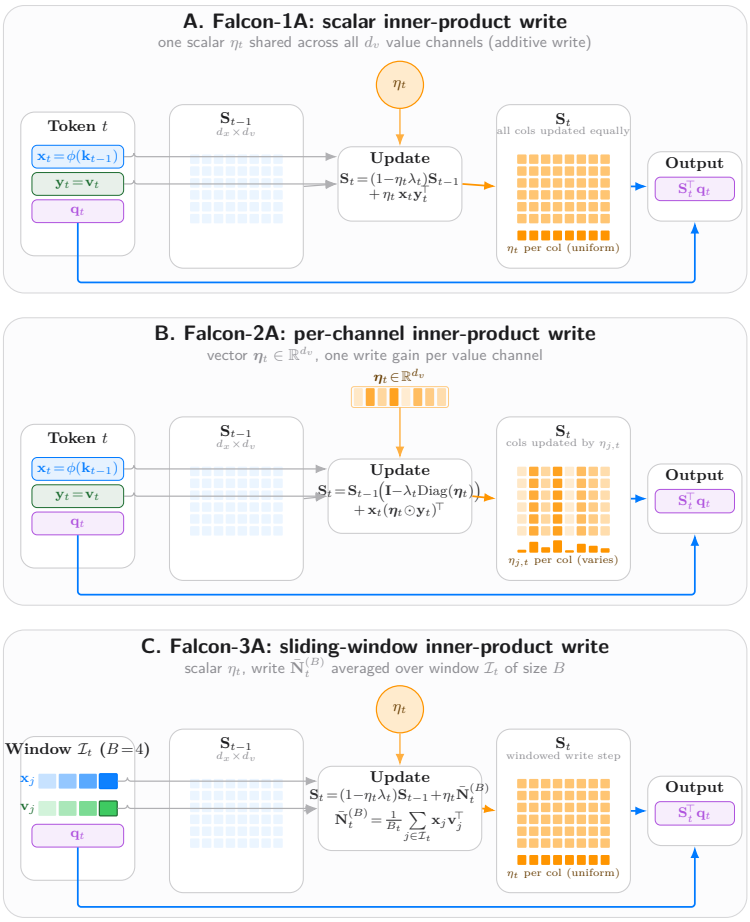

Figure 6 Falcon-1A vs. Falcon-2A vs. Falcon-3A. Each panel shows one inner-product fast-weight update step on the matrix-valued state $\mathbf{S}_{t-1} \in \mathbb{R}^{d_x \times d_v}$. Unlike the regression family (Falcon-1/2/3), the write is purely additive in the target rather than in the residual: there is no $\mathbf{S}_{t-1}^\top \mathbf{x}_t$ subtraction inside the write, only the global shrinkage factor $(1-\eta_t\lambda_t)$ on the carry. **(A) Falcon-1A** applies one scalar η_t to all d_v value channels; every column of $\mathbf{S}_t$ receives the same energy-normalized write gain $\eta_t = \beta_t/(\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon)$ (uniform orange shading and flat per-column bar). **(B) Falcon-2A** promotes the write gain to a vector $\boldsymbol{\eta}_t \in \mathbb{R}^{d_v}$ shown as the orange strip above the update card: each column of $\mathbf{S}_t$ receives its own gain $\eta_{j,t}$ (column shading and bar heights vary), while the write-feature $\mathbf{x}_t$ and target $\mathbf{v}_t$ remain shared. **(C) Falcon-3A** reverts to a scalar η_t but writes the windowed average cross-covariance $\bar{\mathbf{N}}_t^{(B)} = B_t^{-1}\sum_{j\in\mathcal{I}_t} \mathbf{x}_j\mathbf{v}_j^\top$ over a sliding window $\mathcal{I}_t$ of B recent causal pairs; the blue/green strips show four $\{\mathbf{x}_j, \mathbf{v}_j\}_{j\in\mathcal{I}_t}$ entries (light $\rightarrow$ dark = older $\rightarrow$ newer; the boxed cell at $j=t$ is the current step). The denominator $\eta_t = \beta_t/(\bar{E}_t^{(B)} + \lambda_t + \varepsilon)$ uses the window energy $\bar{E}_t^{(B)}$ as a write-magnitude normalizer. The read $\mathbf{o}_t = \mathbf{S}_t^\top \mathbf{q}_t$ uses the updated state in all three cases.

Gradient and update. The gradient of Eq. (4.5) is

$$\nabla_{\mathbf{S}} \ell_t^{\mathrm{ip}}(\mathbf{S}) = -\mathbf{x}_t\mathbf{y}_t^\top + \lambda_t \mathbf{S}.$$

A scalar gradient step gives the Linear-Attention/Mamba-2 style update, which we denote FALCON-1A:

$$\mathbf{S}_t = (1 - \eta_t\lambda_t)\mathbf{S}_{t-1} + \eta_t\,\mathbf{x}_t\mathbf{y}_t^\top, \tag{4.6}$$

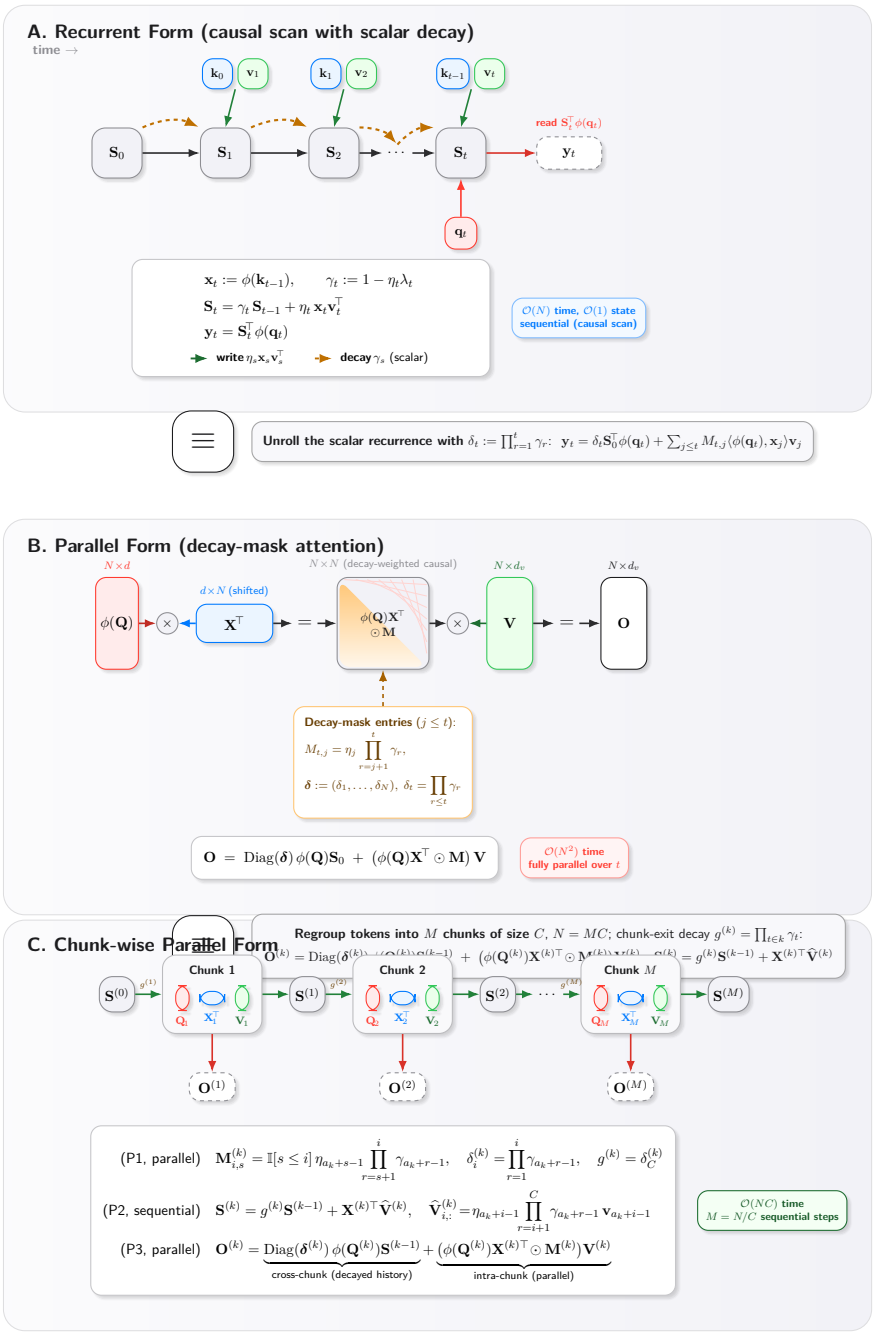

Figure 7 Three equivalent views of Falcon-1A. (A) The recurrent form maintains a bounded matrix state $\mathbf{S}_t \in \mathbb{R}^{d_x \times d_v}$. Compared with linear attention, each carry is multiplied by a scalar decay $\gamma_t = 1 - \eta_t \lambda_t$, and the write feature is the prefix feature $\mathbf{x}_t := \phi(\mathbf{k}_{t-1})$ rather than the same-step feature $\phi(\mathbf{k}_t)$. (B) Unrolling the scalar recurrence yields a masked-attention form with the same shifted features but a *decay-weighted* causal mask: $M_{t,j} = \eta_j \prod_{r=j+1}^{t} \gamma_r$ for $j \le t$ and 0 otherwise. (C) The chunk-wise form splits the sequence into M chunks of size C.

where $\mathbf{x}_t = \phi(\mathbf{k}_{t-1})$ and $\mathbf{y}_t = \mathbf{v}_t$ under next-latent alignment. Setting $\lambda_t = 0$ recovers the usual additive write $\mathbf{S}_t = \mathbf{S}_{t-1} + \eta_t \mathbf{x}_t \mathbf{y}_t^\top$.

Falcon-2A: per-column inner-product write. Because Eq. (4.5) decomposes over value coordinates, we can assign each column its own energy-normalized learning rate. Let $\boldsymbol{\eta}_t = (\eta_{1,t}, \ldots, \eta_{d_v,t})^\top$. The per-column inner-product update is

$$\mathbf{S}_t = \mathbf{S}_{t-1}(\mathbf{I}_{d_v} - \lambda_t \operatorname{Diag}(\boldsymbol{\eta}_t)) + \mathbf{x}_t(\boldsymbol{\eta}_t \odot \mathbf{y}_t)^\top. \tag{4.7}$$

Equivalently, the j-th column evolves as

$$\mathbf{s}_{t,j} = (1 - \eta_{j,t}\lambda_t)\mathbf{s}_{t-1,j} + \eta_{j,t} y_{t,j} \mathbf{x}_t.$$

Thus FALCON-2A is not the scalar inner-product rule; it is the per-column inner-product analogue of FALCON-2.

Falcon-1A/Falcon-2A step sizes. Unlike regression, the inner-product objective is λ_t-smooth independently of the write-feature energy. Accordingly, the objective-matched denominator would depend only on λ_t; in the inner-product implementations studied here, we instead retain an energy-normalized write gain to control the magnitude of the additive write. Let $E_t := \|\mathbf{x}_t\|_2^2$, and if the scale-coupled parameterization is active set $\lambda_t := \bar{\lambda}_t E_t$ before applying the update. The scalar FALCON-1A step size is

$$\eta_t = \frac{\beta_t}{E_t + \lambda_t + \varepsilon}, \qquad \beta_t \in (0, 2),\ \varepsilon \geq 0. \tag{4.8}$$

The per-column FALCON-2A step sizes are

$$\eta_{j,t} = \frac{\beta_{j,t}}{E_t + \lambda_t + \varepsilon}, \qquad \beta_{j,t} \in (0, 2),\ \varepsilon \geq 0. \tag{4.9}$$

As in Eq. (3.4), we set $\eta_t := 0$ (or $\eta_{j,t} := 0$ for all j) when the denominator vanishes, and also at the boundary sentinel $t = 1$ when $\mathbf{x}_1 = \mathbf{0}$ is imposed. When $\lambda_t > 0$, these choices satisfy $\eta_t < 2/\lambda_t$ and $\eta_{j,t} < 2/\lambda_t$ for any admissible $\beta_t, \beta_{j,t}$. Since ℓ_t^{ip} is λ_t-smooth (its Hessian is $\lambda_t \mathbf{I}$), Lemma 3.1 yields per-step descent for the unclamped scalar update, and the same argument applies column-wise to Eq. (4.7). If the later positive-decay clamp is activated for log-space unrolling, the implemented shrinkage should be interpreted as using the effective ridge coefficient $\widetilde{\lambda}_t := \alpha_t/\eta_t$ in the scalar case, or $\widetilde{\lambda}_{j,t} := \alpha_{j,t}/\eta_{j,t}$ in the per-column case, whenever the corresponding step size is positive. The E_t term is not required by curvature, but stabilizes the write magnitude and yields a sensible $\lambda_t \to 0$ limit.

Decay positivity. Some parallel/unrolled forms (Section 4.6) use $\gamma_t := 1 - \eta_t\lambda_t$ in log space and therefore require $\gamma_t > 0$. In implementations, compute $\alpha_t := \eta_t\lambda_t$ and, if necessary, clamp $\alpha_t \leftarrow \min(\alpha_t, 1 - \varepsilon_\gamma)$ before computing $\log \gamma_t = \operatorname{log1p}(-\alpha_t)$ (fp32). The clamp is inactive whenever $\eta_t\lambda_t < 1 - \varepsilon_\gamma$; in that regime, the dynamics match $\gamma_t = 1 - \eta_t\lambda_t$ exactly. As above, the descent statement pertains to the unclamped recurrence. When the clamp activates, the implemented recurrence should be viewed as a numerically safe surrogate whose shrinkage path uses the effective ridge coefficient

$$\widetilde{\lambda}_t := \alpha_t/\eta_t$$

whenever $\eta_t > 0$ (and $\widetilde{\lambda}_t := 0$ when $\eta_t = 0$), while the additive write gain remains η_t.

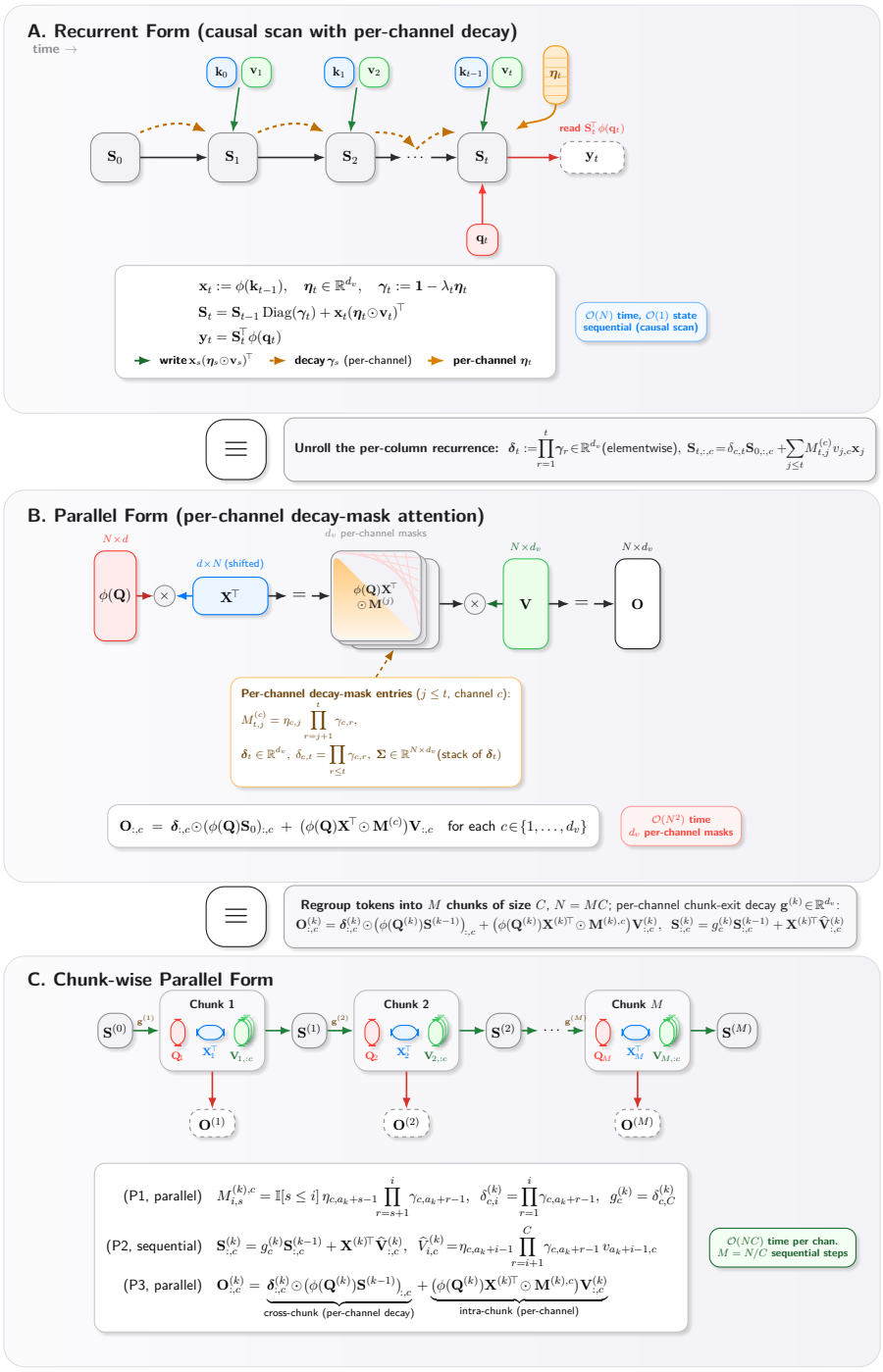

Figure 8 Three equivalent views of Falcon-2A. (A) The recurrent form maintains a bounded matrix state $\mathbf{S}_t \in \mathbb{R}^{d_x \times d_v}$. (B) Unrolling the per-column recurrence yields a masked-attention form with the same shifted features but d_v *channel-specific* decay-weighted causal masks: $M_{t,j}^{(c)} = \eta_{c,j} \prod_{r=j+1}^{t} \gamma_{c,r}$ for $j \le t$ and 0 otherwise. (C) The chunk-wise form splits the sequence into M chunks of size C.

Algorithm 2 Falcon-3 (Recurrent Form)

Require: Query features $\{\mathbf{q}_t\}_{t=1}^T$ with $\mathbf{q}_t = \phi(\mathbf{q}_t)$, shifted write-features $\{\mathbf{x}_t\}_{t=1}^T$ with $\mathbf{x}_1 = \mathbf{0}$ and $\mathbf{x}_t = \phi(\mathbf{k}_{t-1})$ for $t \geq 2$, values $\{\mathbf{v}_t\}_{t=1}^T$, window size B, gains $\beta_t \in (0, 2)$, ridge $\lambda_t \geq 0$, stabilizer $\varepsilon > 0$, init $\mathbf{S}_0 = \mathbf{0}$.

Ensure: Outputs $\mathbf{O} \in \mathbb{R}^{T \times d_v}$ with row t equal to $\mathbf{o}_t^\top = \mathbf{q}_t^\top \mathbf{S}_t$ (read-after-write), final matrix state $\mathbf{S}_T$, and final FIFO buffer $\mathcal{W}_T$; exact continuation across a later segment requires both $\mathbf{S}_T$ and $\mathcal{W}_T$.

Initialize empty FIFO buffer $\mathcal{W} \leftarrow [\,]$ and output buffer $\mathbf{O} \leftarrow \mathbf{0}$.
for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    $\eta_t \leftarrow 0, \quad \mathbf{S}_t \leftarrow \mathbf{S}_{t-1}, \quad \mathbf{o}_t \leftarrow \mathbf{S}_t^\top \mathbf{q}_t$
    Write $\mathbf{o}_t^\top$ into row t of $\mathbf{O}$ and **continue**
  end if
  Push $(\mathbf{x}_t, \mathbf{v}_t)$ into $\mathcal{W}$; if $|\mathcal{W}| > B$, pop the oldest pair.
  $B_t \leftarrow |\mathcal{W}|, \quad \mathbf{U}_t \leftarrow \mathbf{0}_{d_x \times d_v}$
  for all $(\mathbf{x}, \mathbf{v}) \in \mathcal{W}$ **do**
    $\mathbf{U}_t \leftarrow \mathbf{U}_t + \mathbf{x}\big(\mathbf{v} - \mathbf{S}_{t-1}^\top \mathbf{x}\big)^\top$
  end for
  Let $\mathbf{X}_t \in \mathbb{R}^{d_x \times B_t}$ stack the write-features in $\mathcal{W}$.
  $\mu_t^{(B)} \leftarrow \lambda_{\max}(\mathbf{X}_t^\top \mathbf{X}_t)/B_t$ ▷ equiv. $\lambda_{\max}(\bar{\mathbf{C}}_t^{(B)})$
  $\eta_t \leftarrow \beta_t/(\mu_t^{(B)} + \lambda_t + \varepsilon)$
  $\mathbf{S}_t \leftarrow (1 - \eta_t \lambda_t)\mathbf{S}_{t-1} + (\eta_t/B_t)\,\mathbf{U}_t$
  $\mathbf{o}_t \leftarrow \mathbf{S}_t^\top \mathbf{q}_t$
  Write $\mathbf{o}_t^\top$ into row t of $\mathbf{O}$
end for
return $(\mathbf{O}, \mathbf{S}_T, \mathcal{W})$

4.4 Mini-batch Update Rule for Regression (Falcon-3)

To better capture local dependencies and reduce noise accumulation, we introduce a *sliding-state* mechanism: instead of updating the state from only the instantaneous residual, we take a single mini-batch gradient step on a finite history window of nominal size B.

Falcon-3 can be viewed as a sliding-window specialization of the internal-objective view exemplified by ATLAS (Behrouz et al., 2025), but here instantiated with a linear matrix memory, a squared-error objective, and strict next-latent alignment. Exact continuation across segment boundaries additionally requires a fixed-width tail of the last $B-1$ causal pairs; Appendix C.7 records the details.

Sequence-parallel training. After zero-padding each active window to width B, Falcon-3 becomes a fixed-rank-B low-rank recurrence. Algorithm 2 gives the reference sequential update, and Algorithm 3 gives the chunk-parallel ParallelFlow implementation based on the positive-decay reduction in Eq. (4.16). Appendix H records the driver construction and mask conventions. The inner-product counterpart (Falcon-3A) admits an explicit masked linear-attention form (Section 4.6), enabling fully vectorized training over the sequence dimension.

For $t \geq 2$, let the active window indices be $\mathcal{I}_t = \{j \mid \max(2, t - B + 1) \leq j \leq t\}$ and denote the realized window size by $B_t := |\mathcal{I}_t| \leq B$ (so $B_t \geq 1$). Define the write feature $\mathbf{x}_j := \phi(\mathbf{k}_{j-1})$ for $j \geq 2$ (so $\mathbf{x}_j = \mathbf{k}_{j-1}$ when ϕ is the identity), and impose the boundary convention $\mathbf{x}_1 := \mathbf{0}$. We set $\eta_1 := 0$, so the $t = 1$ write is a no-op; all windowed objectives/updates below are defined for $t \geq 2$. To make the update magnitude (and hence the effective decay) invariant to the nominal window size B, we

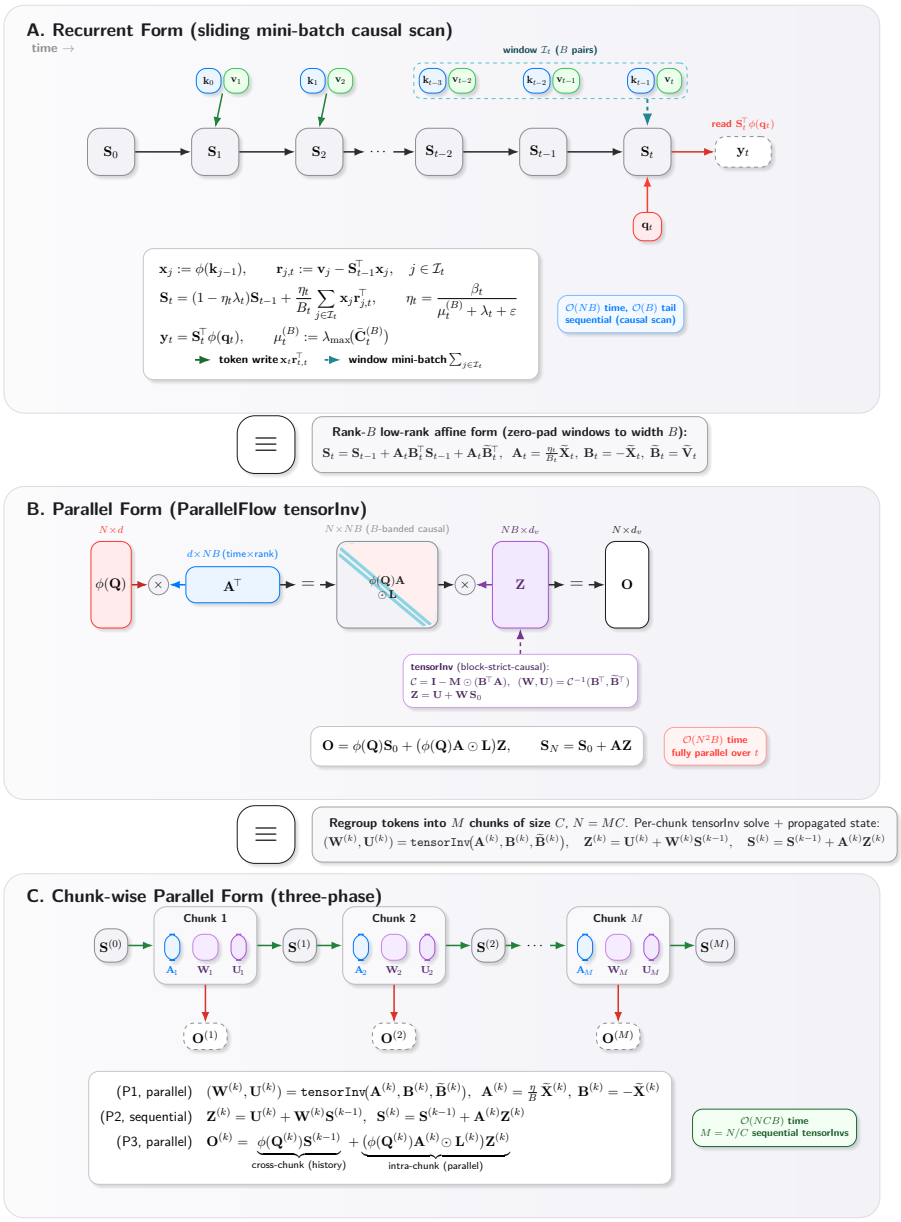

Figure 9 Three equivalent views of Falcon-3. (A) The recurrent form maintains a bounded matrix state $\mathbf{S}_t \in \mathbb{R}^{d_x \times d_v}$. Each step writes a mini-batch gradient on the window $\mathcal{I}_t$ of the last B causal pairs $\{(\phi(\mathbf{k}_{j-1}), \mathbf{v}_j)\}_{j\in\mathcal{I}_t}$, with residuals $\mathbf{r}_{j,t} = \mathbf{v}_j - \mathbf{S}_{t-1}^\top \mathbf{x}_j$ all evaluated at the pre-update state. (B) Zero-padding each active window to width B recasts the recurrence as a rank-B affine update $\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{A}_t \mathbf{B}_t^\top \mathbf{S}_{t-1} + \mathbf{A}_t \widetilde{\mathbf{B}}_t^\top$. (C) Splitting the sequence into M chunks of size C yields the three-phase ParallelFlow algorithm.

optimize the window-average squared loss:

$$\ell_t^{\mathrm{reg},(\mathrm{B})}(\mathbf{S}) := \frac{1}{2B_t} \sum_{j\in\mathcal{I}_t} \|\mathbf{S}^\top \mathbf{x}_j - \mathbf{v}_j\|_2^2 + \frac{\lambda_t}{2}\|\mathbf{S}\|_F^2, \qquad (t \geq 2). \tag{4.10}$$

Sufficient Statistics. We define the sliding covariance $\mathbf{C}_t^{(B)}$ and cross-covariance $\mathbf{N}_t^{(B)}$ matrices:

$$\mathbf{C}_t^{(B)} \triangleq \sum_{j\in\mathcal{I}_t} \mathbf{x}_j\mathbf{x}_j^\top, \qquad \mathbf{N}_t^{(B)} \triangleq \sum_{j\in\mathcal{I}_t} \mathbf{x}_j\mathbf{v}_j^\top. \tag{4.11}$$

Define the window-averaged statistics

$$\bar{\mathbf{C}}_t^{(B)} := \frac{1}{B_t}\mathbf{C}_t^{(B)}, \qquad \bar{\mathbf{N}}_t^{(B)} := \frac{1}{B_t}\mathbf{N}_t^{(B)}.$$

Then the gradient evaluated at the pre-update state is

$$\nabla_{\mathbf{S}}\ell_t^{\mathrm{reg,(B)}}(\mathbf{S}_{t-1}) = \bar{\mathbf{C}}_t^{(B)}\mathbf{S}_{t-1} - \bar{\mathbf{N}}_t^{(B)} + \lambda_t\mathbf{S}_{t-1}.$$

Update Rule. We apply a block-normalized gradient step:

$$\mathbf{S}_t = \mathbf{S}_{t-1} - \eta_t\nabla_{\mathbf{S}}\ell_t^{\mathrm{reg,(B)}}(\mathbf{S}_{t-1}). \tag{4.12}$$

Substituting the gradient yields the affine update

$$\mathbf{S}_t = (\mathbf{I}_{d_x} - \eta_t(\bar{\mathbf{C}}_t^{(B)} + \lambda_t\mathbf{I}_{d_x}))\mathbf{S}_{t-1} + \eta_t\bar{\mathbf{N}}_t^{(B)}. \tag{4.13}$$

Equivalently, collecting residuals at the pre-update state yields

$$\mathbf{S}_t = (1-\eta_t\lambda_t)\mathbf{S}_{t-1} \;+\; \frac{\eta_t}{B_t}\sum_{j\in\mathcal{I}_t}\mathbf{x}_j(\mathbf{v}_j - \mathbf{S}_{t-1}^\top\mathbf{x}_j)^\top. \tag{4.14}$$

This is the direct mini-batch analogue of Eq. (3.3): all window residuals are evaluated at the pre-update state $\mathbf{S}_{t-1}$, and the update averages their rank-one gradients.

For $t \geq 2$, let $\mathbf{X}_t \in \mathbb{R}^{d_x\times B_t}$ stack the active-window write-features, so that $\bar{\mathbf{C}}_t^{(B)} = \mathbf{X}_t\mathbf{X}_t^\top/B_t$. We normalize by the exact local smoothness scale of the windowed ridge objective rather than by its trace upper bound:

$$\mu_t^{(B)} := \lambda_{\max}(\bar{\mathbf{C}}_t^{(B)}) = \frac{\|\mathbf{X}_t\|_2^2}{B_t} = \frac{\lambda_{\max}(\mathbf{X}_t^\top\mathbf{X}_t)}{B_t}, \qquad \eta_t = \frac{\beta_t}{\mu_t^{(B)} + \lambda_t + \varepsilon}, \qquad \beta_t \in (0,2),\ \varepsilon > 0.$$

Then $L_t^{(B)} = \mu_t^{(B)} + \lambda_t$, so whenever $L_t^{(B)} > 0$ this normalization ensures $\eta_t \in (0, 2/L_t^{(B)})$ for any $\beta_t \in (0,2)$, and Lemma 3.1 yields per-step descent for Eq. (4.14) before any positive-decay clamp. If $L_t^{(B)} = 0$ (equivalently, $\bar{\mathbf{C}}_t^{(B)} = 0$ and $\lambda_t = 0$), then $\bar{\mathbf{N}}_t^{(B)} = 0$ as well and the update is a no-op. Crucially, because we optimize the window average, $\bar{\mathbf{N}}_t^{(B)}$ is an average and $\mu_t^{(B)}$ is the spectral norm of an average covariance, so neither quantity grows linearly with the nominal window size B. If the write feature itself is RMS-normalized, then

$$\mu_t^{(B)} \leq \bar{E}_t^{(B)} := \mathrm{tr}(\bar{\mathbf{C}}_t^{(B)}) \approx d_x,$$

so the denominator remains $O(d_x)$ rather than $O(Bd_x)$; for a generic kernel map ϕ, the correct statement is simply that $\mu_t^{(B)}$ tracks the realized windowed smoothness scale. Consequently, neither the injection $\eta_t\bar{\mathbf{N}}_t^{(B)}$ nor the decay fraction $\alpha_t := \eta_t\lambda_t$ is systematically amplified by increasing B. If the scale-coupled ridge parameterization of Section 4.1 is enabled, replace λ_t throughout

this subsection by $\lambda_t^{\mathrm{eff}} := \bar{\lambda}_t \, \mu_t^{(B)}$. In the current implementation, this smoothness statistic can be treated as a statistics-only multiplier when constructing λ_t^{eff} (detached / stop-gradient through the multiplier), while the step-size denominator still uses the live $\mu_t^{(B)}$. Importantly, one need not materialize the $d_x \times d_x$ matrix $\mathbf{C}_t^{(B)}$ to evaluate either the gradient or the step size: if we stack the window write-features into $\mathbf{X}_t \in \mathbb{R}^{d_x \times B_t}$, then

$$\bar{\mathbf{C}}_t^{(B)} \mathbf{S}_{t-1} = \frac{1}{B_t} \mathbf{X}_t (\mathbf{X}_t^\top \mathbf{S}_{t-1}), \qquad \mu_t^{(B)} = \frac{\lambda_{\max}(\mathbf{X}_t^\top \mathbf{X}_t)}{B_t}. \tag{4.15}$$

Since $B_t \le B$ is small, $\mu_t^{(B)}$ can be computed exactly from the $B_t \times B_t$ Gram matrix or approximated with a few power iterations. For the implemented positive-decay recurrence, define

$$\gamma_t := 1 - \alpha_t, \qquad \alpha_t^{\mathrm{raw}} := \eta_t \lambda_t, \qquad \alpha_t := \min(\alpha_t^{\mathrm{raw}}, \, 1 - \varepsilon_\gamma), \qquad c_0 := 1, \qquad c_t := \prod_{r=1}^{t} \gamma_r, \qquad \widetilde{\mathbf{S}}_t := \mathbf{S}_t / c_t.$$

When $\alpha_t = \alpha_t^{\mathrm{raw}}$, this is exactly the ridge-gradient recurrence in Eq. (4.14). If the clamp activates, the implemented shrinkage path should instead be interpreted as using the effective coefficient

$$\widetilde{\lambda}_t := \begin{cases} \alpha_t / \eta_t, & \eta_t > 0, \\ 0, & \eta_t = 0, \end{cases}$$

while keeping the same residual injection gain η_t. Thus the descent claim above applies to the unclamped update; the clamped update is a positive-decay surrogate. With this convention, the implemented recurrence is equivalent to

$$\widetilde{\mathbf{S}}_t = \widetilde{\mathbf{S}}_{t-1} + \frac{\widehat{\eta}_t}{B_t} \sum_{j \in \mathcal{I}_t} \mathbf{x}_j \Big(\frac{\mathbf{v}_j}{c_{t-1}} - \widetilde{\mathbf{S}}_{t-1}^\top \mathbf{x}_j \Big)^\top, \qquad \widehat{\eta}_t := \eta_t / \gamma_t. \tag{4.16}$$

Algorithm 3 realizes Eq. (4.16) chunk-locally via log-prefix decays; Appendix C.7 records the corresponding continuation and chunk-boundary details.

Algorithm 3 Falcon-3 via ParallelFlow (Chunk-parallel Forward)

Require: Queries $\{\mathbf{q}_t\}_{t=1}^T$, keys $\{\mathbf{k}_t\}_{t=1}^T$, values $\{\mathbf{v}_t\}_{t=1}^T$, feature map ϕ (default identity), window size B, gains $\beta_t \in (0, 2)$, ridge $\lambda_t \ge 0$, stabilizer $\varepsilon > 0$, decay clamp $\varepsilon_\gamma > 0$, chunk size C (assume $C \mid T$), init $\mathbf{S}_0$.

Ensure: Outputs $\mathbf{O} \in \mathbb{R}^{T \times d_v}$ with row t equal to $\mathbf{o}_t^\top = \phi(\mathbf{q}_t)^\top \mathbf{S}_t$ (read-after-write), final matrix state $\mathbf{S}_T$, and boundary tail $\mathcal{T}_T := \{(\mathbf{x}_j, \mathbf{v}_j)\}_{j=\max(2, T-B+2)}^T$ for exact continuation across a later segment.

Write/query features: set $\mathbf{x}_1 \leftarrow \mathbf{0}$, for $t \ge 2$ set $\mathbf{x}_t \leftarrow \phi(\mathbf{k}_{t-1})$, and for all t set $\mathbf{q}_t \leftarrow \phi(\mathbf{q}_t)$.

Window indices $\mathcal{I}_t \leftarrow \{j \mid \max(2, t - B + 1) \le j \le t\}$, window sizes $B_t \leftarrow |\mathcal{I}_t|$, and $B_t^+ \leftarrow \max(1, B_t)$.

Window smoothness statistics: $\mu_1^{(B)} \leftarrow 0$ and for $t \ge 2$ set $\mu_t^{(B)} \leftarrow \lambda_{\max}(\mathbf{G}_t)/B_t$, where $\mathbf{G}_t := \mathbf{X}_t^\top \mathbf{X}_t$ and $\mathbf{X}_t := [\mathbf{x}_j]_{j \in \mathcal{I}_t} \in \mathbb{R}^{d_x \times B_t}$.

Step sizes: set $\eta_1 \leftarrow 0$ and for $t \ge 2$ set $\eta_t \leftarrow \beta_t / (\mu_t^{(B)} + \lambda_t + \varepsilon)$.

Clamp decays: $\alpha_t \leftarrow \min(\eta_t \lambda_t, \, 1 - \varepsilon_\gamma)$, $\log \gamma_t \leftarrow \mathrm{log1p}(-\alpha_t)$, $\gamma_t \leftarrow \exp(\log \gamma_t)$, and $\widehat{\eta}_t \leftarrow \eta_t / \gamma_t$.

Partition into $M = T/C$ chunks $[a_k, b_k] = [(k-1)C+1, kC]$.

▷ **Phase 1: Intra-chunk ParallelFlow solves (parallel over chunks)**

for $k = 1$ **to** M **in parallel do**

  $a \leftarrow a_k$, $b \leftarrow b_k$, $p \leftarrow \max(2, a - B + 1)$, $J \leftarrow \{p, \ldots, b\}$.

  Slice $\mathbf{Q}^{(k)} \leftarrow (\mathbf{q}_t^\top)_{t=a}^b \in \mathbb{R}^{C \times d_x}$ and gather the write pairs $\{(\mathbf{x}_j, \mathbf{v}_j)\}_{j \in J}$.

Build zero-padded window stacks $\mathbf{X}_{\text{win}}^{(k)} \in \mathbb{R}^{d_x \times (CB)}$ and $\mathbf{V}_{\text{win}}^{(k)} \in \mathbb{R}^{d_v \times (CB)}$: the i-th block of B columns contains the pairs from $\mathcal{I}_{a+i-1}$ (in increasing j), padded with $\mathbf{0}$ to width B.

Local log-prefix decays (length-C scan): $\delta_0^{(k)} \leftarrow 1$, $u_0 \leftarrow 0$, and for $i = 1{:}C$ set $u_i \leftarrow u_{i-1} + \log \gamma_{a+i-1}$, $\delta_i^{(k)} \leftarrow \exp(u_i)$.

Rescale values blockwise: divide every column in block i of $\mathbf{V}_{\text{win}}^{(k)}$ by the scalar $\delta_{i-1}^{(k)}$, yielding $\widehat{\mathbf{V}}_{\text{win}}^{(k)}$.

▷ Flatten time×rank. Same-time rank components are uncoupled; only earlier time-blocks interact inside `tensorInv`.

Driver matrices: scale block i of $\mathbf{X}_{\text{win}}^{(k)}$ by $s_i := \widehat{\eta}_{a+i-1}/B_{a+i-1}^{+}$ to obtain $\mathbf{A}^{(k)} \in \mathbb{R}^{d_x \times (CB)}$; set $\mathbf{R}^{(k)} \leftarrow -\mathbf{X}_{\text{win}}^{(k)}$ and $\widetilde{\mathbf{R}}^{(k)} \leftarrow \widehat{\mathbf{V}}_{\text{win}}^{(k)}$.

$(\mathbf{W}^{(k)}, \mathbf{U}^{(k)}) \leftarrow \texttt{tensorInv}\big(\mathbf{A}^{(k)}, \mathbf{R}^{(k)}, \widetilde{\mathbf{R}}^{(k)}\big)$ ▷ block-strict-causal solve on (time, rank) pairs

Optional scan form: $\mathbf{M}^{(k)} \leftarrow \delta_C^{(k)}\big(\mathbf{I}_{d_x} + \mathbf{A}^{(k)}\mathbf{W}^{(k)}\big)$ and $\mathbf{b}^{(k)} \leftarrow \delta_C^{(k)}\mathbf{A}^{(k)}\mathbf{U}^{(k)}$.

Cache $\big(\mathbf{Q}^{(k)}, \mathbf{A}^{(k)}, \mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \boldsymbol{\delta}^{(k)}\big)$ where $\boldsymbol{\delta}^{(k)} := (\delta_1^{(k)}, \ldots, \delta_C^{(k)})$.

end for

▷ **Phase 2: Inter-chunk boundary propagation (sequential; equivalently an associative scan over $(\mathbf{M}^{(k)}, \mathbf{b}^{(k)})$)**

$\mathbf{S}_{\text{in}}^{(1)} \leftarrow \mathbf{S}_0$.

for $k = 1$ **to** M **do**

Store $\mathbf{S}_{\text{in}}^{(k)}$.

$\mathbf{Z}^{(k)} \leftarrow \mathbf{U}^{(k)} + \mathbf{W}^{(k)}\mathbf{S}_{\text{in}}^{(k)}$.

$\widehat{\mathbf{S}}_{\text{out}}^{(k)} \leftarrow \mathbf{S}_{\text{in}}^{(k)} + \mathbf{A}^{(k)}\mathbf{Z}^{(k)}$. ▷ chunk exit in the local renormalized domain

$\mathbf{S}_{\text{in}}^{(k+1)} \leftarrow \delta_C^{(k)}\,\widehat{\mathbf{S}}_{\text{out}}^{(k)}$. ▷ restore original scale at the chunk boundary; equiv. $\mathbf{S}_{\text{in}}^{(k+1)} = \mathbf{M}^{(k)}\mathbf{S}_{\text{in}}^{(k)} + \mathbf{b}^{(k)}$

end for

$\mathbf{S}_T \leftarrow \mathbf{S}_{\text{in}}^{(M+1)}$.

▷ **Phase 3: Materialize token outputs (parallel over chunks)**

for $k = 1$ **to** M **in parallel do**

$\mathbf{Z}^{(k)} \leftarrow \mathbf{U}^{(k)} + \mathbf{W}^{(k)}\mathbf{S}_{\text{in}}^{(k)}$.

$\mathbf{H}^{(k)} \leftarrow \mathbf{Q}^{(k)}\mathbf{S}_{\text{in}}^{(k)}$ ▷ $C \times d_v$

$\mathbf{P}^{(k)} \leftarrow \mathbf{Q}^{(k)}\mathbf{A}^{(k)}$ ▷ $C \times (CB)$

Define the block-causal mask $L^{(k)} \in \{0,1\}^{C \times (CB)}$ by $L^{(k)}_{i,(m-1)B+1:mB} = \mathbb{I}[m \le i]\mathbf{1}_{1\times B}$ (read-after-write).

$\widehat{\mathbf{O}}^{(k)} \leftarrow \mathbf{H}^{(k)} + \big(\mathbf{P}^{(k)} \odot L^{(k)}\big)\mathbf{Z}^{(k)}$. ▷ local-renormalized outputs

Row-rescale: $\mathbf{O}_{i,:}^{(k)} \leftarrow \delta_i^{(k)}\,\widehat{\mathbf{O}}_{i,:}^{(k)}$ for $i = 1{:}C$, and write into rows $a_k{:}b_k$ of $\mathbf{O}$.

end for

$\mathcal{T}_T \leftarrow \{(\mathbf{x}_j, \mathbf{v}_j)\}_{j=\max(2,T-B+2)}^{T}$.

return $(\mathbf{O}, \mathbf{S}_T, \mathcal{T}_T)$.

Boundary/scan details. The offline overlap interpretation, exact continuation requirement, and explicit associative chunk map are given in Appendix C.7 and Appendix H.1.

4.5 Mini-batch Update Rule for Inner Product Loss (Falcon-3A)

We apply the same sliding-window principle to the inner-product objective, while keeping the same parameter semantics as in the rest of the family: β_t is the dimensionless gain, λ_t is the actual shrinkage coefficient used by the recurrence (after any optional scale coupling), and η_t is the resulting normalized step size.

For $t \geq 2$, let $B_t := |\mathcal{I}_t| \leq B$ (and skip the boundary update at $t = 1$). The windowed loss is

$$\ell_t^{\mathrm{ip,(B)}}(\mathbf{S}) := -\frac{1}{B_t} \sum_{j \in \mathcal{I}_t} \langle \mathbf{S}^\top \mathbf{x}_j, \mathbf{v}_j \rangle + \frac{\lambda_t}{2} \|\mathbf{S}\|_F^2. \tag{4.17}$$

Define the window-averaged cross-covariance and write energy

$$\bar{\mathbf{N}}_t^{(B)} := \frac{1}{B_t} \sum_{j \in \mathcal{I}_t} \mathbf{x}_j \mathbf{v}_j^\top, \qquad \bar{E}_t^{(B)} := \frac{1}{B_t} \sum_{j \in \mathcal{I}_t} \|\mathbf{x}_j\|_2^2.$$

Then the gradient evaluated at the pre-update state is

$$\nabla_{\mathbf{S}} \ell_t^{\mathrm{ip,(B)}}(\mathbf{S}_{t-1}) = -\bar{\mathbf{N}}_t^{(B)} + \lambda_t \mathbf{S}_{t-1}.$$

Unlike FALCON-3, whose regression step size uses the local smoothness $\mu_t^{(B)}$, the inner-product windowed rule uses the window-energy statistic as a practical write-gain normalizer. Let $E_t^{(B)} := \bar{E}_t^{(B)}$, and if the scale-coupled parameterization is active set $\lambda_t := \bar{\lambda}_t E_t^{(B)}$ before computing the step size:

$$\eta_t = \frac{\beta_t}{E_t^{(B)} + \lambda_t + \varepsilon}, \qquad \beta_t \in (0, 2),\ \varepsilon > 0, \tag{4.18}$$

with the boundary convention $\eta_1 := 0$. Applying one gradient step gives

$$\mathbf{S}_t = (1 - \eta_t \lambda_t) \mathbf{S}_{t-1} + \eta_t \bar{\mathbf{N}}_t^{(B)}. \tag{4.19}$$

When $\lambda_t > 0$, the objective is λ_t-smooth, so Eq. (4.18) implies $\eta_t < 2/\lambda_t$ for any $\beta_t \in (0, 2)$, and Lemma 3.1 yields per-step descent in $\ell_t^{\mathrm{ip,(B)}}$ for the unclamped update. If the positive-decay clamp below activates, the implemented shrinkage should again be interpreted as using the effective ridge coefficient $\widetilde{\lambda}_t := \alpha_t / \eta_t$ when $\eta_t > 0$ (and 0 when $\eta_t = 0$), rather than as an exact gradient step for the original λ_t. When $\lambda_t = 0$, the objective is linear and unbounded below, so the same normalization should be interpreted as a magnitude stabilizer for the additive write rather than as a bounded-objective guarantee.

For log-space unrolling we introduce only the derived quantities

$$\alpha_t^{\mathrm{raw}} := \eta_t \lambda_t, \qquad \alpha_t := \min(\alpha_t^{\mathrm{raw}}, 1 - \varepsilon_\gamma), \qquad \gamma_t := 1 - \alpha_t \in [\varepsilon_\gamma, 1], \tag{4.20}$$

with $\alpha_1 := 0$ and $\gamma_1 := 1$. Thus α_t and γ_t are implementation variables derived from the same $(\beta_t, \lambda_t, \eta_t)$ parameterization used everywhere else; they are not separate learned controls.

Because we optimize the window average, neither the typical scale of $\bar{\mathbf{N}}_t^{(B)}$ nor the decay fraction $\alpha_t = \eta_t \lambda_t$ grows systematically with the nominal window size B. We again set $\eta_t := 0$ when the denominator in Eq. (4.18) vanishes (in practice we take $\varepsilon > 0$). Appendix C.7 records the stationary calculation and the exact boundary-state requirements. For $B = 1$ and $t \geq 2$, Eq. (4.19) reduces to the scalar non-sliding inner-product update, i.e. FALCON-1A; in the additive ablation $\lambda_t \equiv 0$, the write is purely additive. The per-column non-sliding analogue is FALCON-2A in Eq. (4.7).

4.6 Parallel (Attention) Form of Mini-batch Inner Product Update

Algorithm 4 Chunk-parallel masked attention for Falcon-3A (Forward)

Require: Query features $\mathbf{Q} \in \mathbb{R}^{L\times d_x}$ (row t is $\phi(\mathbf{q}_t)^\top$), write features $\mathbf{X} \in \mathbb{R}^{L\times d_x}$ (row t is $\mathbf{x}_t^\top = \phi(\mathbf{k}_{t-1})^\top$), values $\mathbf{V} \in \mathbb{R}^{L\times d_v}$ (row t is $\mathbf{v}_t^\top$), gains $\boldsymbol{\beta} \in (0,2)^L$, actual ridge coefficients $\boldsymbol{\lambda} \in \mathbb{R}^L_{\geq 0}$, stabilizer $\varepsilon > 0$, window size B, chunk size C (assume $C \mid L$), decay floor $\varepsilon_\gamma > 0$, initial state $\mathbf{S}_{\text{init}} \in \mathbb{R}^{d_x\times d_v}$.

Ensure: Outputs $\mathbf{O} \in \mathbb{R}^{L\times d_v}$ where row t is $\mathbf{o}_t^\top = \phi(\mathbf{q}_t)^\top \mathbf{S}_t$ under $\eta_1 = 0$, $\gamma_1 = 1$, $\bar{\mathbf{N}}_1^{(B)} = \mathbf{0}$, $\bar{E}_1^{(B)} = 0$, and for $t \geq 2$, $\mathbf{S}_t = \gamma_t \mathbf{S}_{t-1} + \eta_t \bar{\mathbf{N}}_t^{(B)}$ with $\bar{\mathbf{N}}_t^{(B)} = \frac{1}{B_t}\sum_{j\in\mathcal{I}_t} \mathbf{x}_j \mathbf{v}_j^\top$ and $\bar{E}_t^{(B)} = \frac{1}{B_t}\sum_{j\in\mathcal{I}_t} \|\mathbf{x}_j\|_2^2$, $\eta_t = \beta_t/(\bar{E}_t^{(B)} + \lambda_t + \varepsilon)$, and $\gamma_t := 1 - \alpha_t$ with $\alpha_t := \min(\eta_t\lambda_t, 1-\varepsilon_\gamma)$.

Set $\mathcal{I}_1 \leftarrow \emptyset$, $B_1 \leftarrow 0$. For $t \geq 2$, define $\mathcal{I}_t = \{j \mid \max(2, t-B+1) \leq j \leq t\}$ and set $B_t \leftarrow |\mathcal{I}_t|$.

When forming window weights, use $B_t^+ \leftarrow \max(1, B_t)$ so that $\mathbb{I}[j \in \mathcal{I}_t]/B_t^+$ is always well-defined.

Set $\bar{E}_1^{(B)} \leftarrow 0$, and for $t \geq 2$ compute $\bar{E}_t^{(B)} \leftarrow \frac{1}{B_t}\sum_{j\in\mathcal{I}_t} \|\mathbf{x}_j\|_2^2$.

If the scale-coupled ridge parameterization is active, set $\lambda_1 \leftarrow 0$ and for $t \geq 2$ set $\lambda_t \leftarrow \bar{\lambda}_t \bar{E}_t^{(B)}$; otherwise treat the provided λ_t as the actual coefficients.

Set $\eta_1 \leftarrow 0$. For $t \geq 2$, compute $\eta_t \leftarrow \beta_t/(\bar{E}_t^{(B)} + \lambda_t + \varepsilon)$.

Set $\alpha_1 \leftarrow 0$, $\gamma_1 \leftarrow 1$, and $\log\gamma_1 \leftarrow 0$. For $t \geq 2$, compute $\alpha_t \leftarrow \min(\eta_t\lambda_t, 1-\varepsilon_\gamma)$, $\log\gamma_t \leftarrow \mathrm{log1p}(-\alpha_t)$, and $\gamma_t \leftarrow \exp(\log\gamma_t)$.

Partition into $M = L/C$ chunks with indices $[a_k, b_k] = [(k-1)C+1, kC]$.

Initialize output buffer $\mathbf{O} \leftarrow \mathbf{0}_{L\times d_v}$.

▷ **Phase 1: Intra-chunk precomputation (parallel)**

for $k = 1$ **to** M **in parallel do**

  $a \leftarrow a_k$, $b \leftarrow b_k$, $p \leftarrow \max(2, a-B+1)$, and let $J := \{p, p+1, \ldots, b\}$ (overlap length $|J| \leq C+B-1$).

  Slice $\mathbf{Q}^{(k)} \leftarrow \mathbf{Q}_{a:b}$, $\mathbf{X}_{\text{ext}}^{(k)} \leftarrow \mathbf{X}_J$, $\mathbf{V}_{\text{ext}}^{(k)} \leftarrow \mathbf{V}_J$, $\boldsymbol{\eta}^{(k)} \leftarrow \boldsymbol{\eta}_{a:b}$, and $\log\boldsymbol{\gamma}^{(k)} \leftarrow (\log\boldsymbol{\gamma})_{a:b}$.

  Compute log-prefix decays inside the chunk: $u_0 \leftarrow 0$ and $u_i \leftarrow u_{i-1} + \log\gamma_i^{(k)}$ for $i = 1, \ldots, C$ (scan); set $\delta_i \leftarrow \exp(u_i)$ and $g^{(k)} \leftarrow \delta_C$.

  Build the causal decay matrix $D^{(k)} \in \mathbb{R}^{C\times C}$ with $D_{i,s}^{(k)} = \mathbb{I}[s \leq i]\exp(u_i - u_s)$.

  Build the B-banded window operator $A^{(k)} \in \mathbb{R}^{C\times|J|}$ (columns indexed by $j \in J$): $A_{s,j}^{(k)} = \mathbb{I}[\, j \in \mathcal{I}_{a+s-1}\,]/B_{a+s-1}^+$.

  Local mask $M^{(k)} \leftarrow D^{(k)}\,\mathrm{Diag}(\boldsymbol{\eta}^{(k)})\,A^{(k)} \in \mathbb{R}^{C\times|J|}$.

  Intra-chunk output $\mathbf{O}_{\text{intra}}^{(k)} \leftarrow \big((\mathbf{Q}^{(k)}{\mathbf{X}_{\text{ext}}^{(k)}}^\top) \odot M^{(k)}\big)\mathbf{V}_{\text{ext}}^{(k)}$.

  Chunk bias for boundary propagation: $m_{\text{end}}^{(k)} \leftarrow M_{C,:}^{(k)}$ and $\mathbf{B}^{(k)} \leftarrow {\mathbf{X}_{\text{ext}}^{(k)}}^\top\big(\mathrm{Diag}(m_{\text{end}}^{(k)})\mathbf{V}_{\text{ext}}^{(k)}\big)$.

  Cache $g^{(k)}$, $\mathbf{B}^{(k)}$, $\boldsymbol{\delta}^{(k)} := (\delta_1, \ldots, \delta_C)$, and $\mathbf{O}_{\text{intra}}^{(k)}$.

end for

▷ **Phase 2: Inter-chunk recurrence (sequential over chunks or scan)**

$\mathbf{S}_{\text{in}}^{(1)} \leftarrow \mathbf{S}_{\text{init}}$

for $k = 1$ **to** M **do**

  Store $\mathbf{S}_{\text{in}}^{(k)}$ ▷ Needed by Phase 3 to materialize the decayed-history term

  $\mathbf{S}_{\text{out}}^{(k)} \leftarrow g^{(k)}\mathbf{S}_{\text{in}}^{(k)} + \mathbf{B}^{(k)}$

  $\mathbf{S}_{\text{in}}^{(k+1)} \leftarrow \mathbf{S}_{\text{out}}^{(k)}$

end for

▷ **Phase 3: Materialize outputs (parallel)**

for $k = 1$ **to** M **in parallel do**

  $H^{(k)} \leftarrow \mathbf{Q}^{(k)}\mathbf{S}_{\text{in}}^{(k)}$ ▷ $C \times d_v$

  $\mathbf{O}_{\text{hist}}^{(k)} \leftarrow \mathrm{Diag}(\boldsymbol{\delta}^{(k)})\,H^{(k)}$

  $\mathbf{O}^{(k)} \leftarrow \mathbf{O}_{\text{hist}}^{(k)} + \mathbf{O}_{\text{intra}}^{(k)}$

Write $\mathbf{O}_{a_k:b_k} \leftarrow \mathbf{O}^{(k)}$.
end for
return $\mathbf{O}$ (concatenate $\{\mathbf{O}^{(k)}\}_{k=1}^{M}$).

We show that the recurrent sliding-window update in Section 4.5 can be written as the sum of (i) a decayed-history term from the incoming boundary state and (ii) an (unnormalized) dot-product attention matrix with a structured causal mask. This view enables GPU-parallel training and matches the usual unrolling of gated linear recurrences into masked attention (Sun et al., 2023; Qin et al., 2024; Yang et al., 2023; Gu and Dao, 2023; Dao and Gu, 2024).

For the unclamped exposition, let λ_s denote the actual shrinkage coefficient after any optional scale coupling, and define

$$\eta_s := \frac{\beta_s}{\bar{E}_s^{(B)} + \lambda_s + \varepsilon}, \qquad \gamma_s := 1 - \eta_s \lambda_s, \qquad \delta_t := \prod_{r=1}^{t} \gamma_r.$$

Then the recurrence $\mathbf{S}_s = \gamma_s \mathbf{S}_{s-1} + \eta_s \bar{\mathbf{N}}_s^{(B)}$ unrolls to

$$\mathbf{S}_t = \delta_t \mathbf{S}_0 + \sum_{j=2}^{t} M_{t,j}\, \mathbf{x}_j \mathbf{v}_j^\top,$$

where, under the boundary convention $\mathbf{x}_1 := \mathbf{0}$ and $\eta_1 = 0$,

$$M_{t,j} := \sum_{s=j}^{\min(t, j+B-1)} \frac{\eta_s}{B_s} \prod_{r=s+1}^{t} \gamma_r, \qquad 2 \le j \le t,$$

and $M_{t,j} := 0$ otherwise. Hence the read-after-write output is

$$\mathbf{o}_t = \delta_t\, \mathbf{S}_0^\top \phi(\mathbf{q}_t) \;+\; \sum_{j=2}^{t} M_{t,j}\, \langle \phi(\mathbf{q}_t), \mathbf{x}_j \rangle \mathbf{v}_j.$$

Equivalently, stacking query features $\mathbf{Q} \in \mathbb{R}^{L \times d_x}$, write features $\mathbf{X} \in \mathbb{R}^{L \times d_x}$, values $\mathbf{V} \in \mathbb{R}^{L \times d_v}$, and mask $\mathbf{M} \in \mathbb{R}^{L \times L}$ with entries $\mathbf{M}_{t,j} = M_{t,j}$, we obtain

$$\mathbf{O} = \mathrm{Diag}(\boldsymbol{\delta})\, \mathbf{Q}\mathbf{S}_0 \;+\; (\mathbf{Q}\mathbf{X}^\top \odot \mathbf{M})\mathbf{V}, \qquad \boldsymbol{\delta} := (\delta_1, \ldots, \delta_L)^\top.$$

For the fresh-sequence default $\mathbf{S}_0 = \mathbf{0}$, only the masked-attention term remains. Appendix D.1 gives the full derivation, the stationary special case, and the chunk-local log-space evaluation used by Algorithm 4.

Backward pass. The numerically stable backward pass is given in Appendix D.2; it differentiates through the structured mask, chunk-local log-decay renormalization, and normalized step-size computation.

Relation to prior internal-objective views. A brief comparison to test-time training and Titans/ATLAS-style internal memory objectives is moved to Appendix C.4.

5 Experiments

Our empirical evaluation focuses on the scalar and sliding inner-product variants, FALCON-1A/FALCON-3A, in 124M-130M-parameter language models. All models are trained on FineWeb-Edu (Penedo et al., 2024) with a matched 50B-token budget and are evaluated by held-out perplexity and downstream task accuracy. We also use variable-length multi-digit addition as a controlled diagnostic for causal storage and length extrapolation. The per-column inner-product rule FALCON-2A is defined in Section 4.3 but is not separately benchmarked in the main tables. The regression variants FALCON-1/FALCON-2/FALCON-3 are developed in Sections 4.2 and 4.4 but are not benchmarked in the main tables. Unless stated otherwise, fast-weight models use the scaled formulation of Section 4.1 and the next-latent boundary convention $\mathbf{x}_1 := \mathbf{0}$. QK-RMSNorm and QK-$\ell_2$ denote the query/key normalization used inside the fast-weight block.

5.1 Language modeling experiments

We train 124M-130M-parameter models for 100,000 optimization steps with sequence length 1,024 and global batch size 480, for a total budget of approximately 49.2B tokens. The Transformer baseline uses a LLaMA-style architecture with RoPE and SwiGLU. Recurrent baselines include RetNet/LightningAttn, Mamba-2, DeltaNet, and Gated DeltaNet. Tables 1 and 2 report the scalar FALCON-1A variants and include the FALCON-3A.3 checkpoint for reference. Unless explicitly ablated, fast-weight models use QK-RMSNorm and lightweight short convolutions on the attention projections. Full optimization and implementation details are deferred to Appendix B. We report teacher-forced perplexity and zero-shot / one-shot downstream accuracy.

On validation-set perplexity (Table 1), the evaluated inner-product variants are competitive but do not uniformly beat the strongest baselines. Among our variants, FALCON-1A.3 (QK-RMSNorm, ctxη-ctxλ) achieves the best perplexity, reaching 17.40 on FineWeb-Edu validation, while Gated DeltaNet is strongest overall in this comparison. Table 2 shows that this family also transfers reasonably well to downstream evaluation: FALCON-1A.2 achieves the best zero-shot average among the listed models, and its one-shot average remains competitive with Mamba-2 and the Transformer. Within the evaluated FALCONfamily, QK-RMSNorm improves perplexity relative to QK-ℓ_2 normalization, while ctxη improves average downstream accuracy over the corresponding ctxβ variant. The main empirical takeaway is therefore not a uniform win at this scale, but that autoregressively aligned normalized inner-product updates preserve competitive language model quality under a realistic pretraining budget.

5.2 Variable-length arithmetic addition tasks

We use variable-length multi-digit addition as a controlled stress test for causal storage and extrapolation, following Kaiser and Sutskever (2015). Each sample presents an n-digit prompt

$$s_{\text{in}} := \texttt{"}\text{digits}_n(a)\ \texttt{+}\ \text{digits}_n(b)\ \texttt{="},$$

and asks the model to generate the reversed $(n{+}1)$-digit sum

$$s_{\text{out}} := \texttt{"}\ \text{rev}\left(\text{digits}_{n+1}(a+b)\right)\texttt{"},$$

so that the least significant digit is produced first. We train on widths sampled uniformly from $\{1, \ldots, 32\}$ and optimize masked next-token log-likelihood on the target suffix only.

Table 1 Comparison of models with 124M-130M parameters trained for a 50B-token budget on FineWeb-Edu. Lower is better. Boldface marks the best recurrent model in each column.

Model	Perplexity ↓		
	Wiki.	LMB.	FineEdu.
124M-130M parameters			
(124M) Transformer (w. RoPE)	33.25	47.43	17.38
(130M) RetNet/LightningAttn	36.86	65.16	18.79
(130M) Mamba-2	34.53	48.74	17.70
(130M) DeltaNet	34.19	52.84	17.84
(130M) Gated DeltaNet	**30.99**	**46.70**	**17.32**
Ours			
(130M) **Falcon-1A.1 (QK-ℓ_2-norm, ctxβ-ctxλ)**	34.41	47.93	17.70
(130M) **Falcon-1A.2 (QK-ℓ_2-norm, ctxη-ctxλ)**	34.20	51.01	17.70
(130M) **Falcon-1A.3 (QK-RMSNorm, ctxη-ctxλ)**	34.02	49.84	17.40
(130M) **Falcon-3A.3 (QK-RMSNorm, ctxη-ctxλ)**	34.08	53.38	17.59

Table 3 reports in-distribution validation accuracy and out-of-distribution teacher-forced target-suffix accuracy averaged over 33–48 digits. FALCON-3A.3 achieves the best extrapolation performance, with 87.2 mean accuracy, followed by FALCON-1A.3 at 85.9. Both outperform the baselines reported here, including RetNet/LightningAttn and the Transformer. We view this experiment as supporting evidence rather than the paper's primary result: it isolates the memory-writing behavior of the recurrent state and shows that the proposed shifted, normalized updates extrapolate well when storage and carry propagation dominate.

6 Related Work

Efficient Sequence Modeling and State-Space Duality. Standard Transformers (Vaswani et al., 2017) incur quadratic $\mathcal{O}(N^2)$ compute and memory costs due to the materialization of the attention matrix. To address this, Linear Attention (Katharopoulos et al., 2020) reorders the matrix multiplication via kernel feature maps, effectively treating the context as a recurrent accumulation of outer products. Other subquadratic attention replacements include random-feature approximations to softmax attention, such as Performer (Choromanski et al., 2020), long-convolution operators such as Hyena (Poli et al., 2023), and retention-style recurrent attention (Sun et al., 2023) or RNN–Transformer hybrids (Peng et al., 2023). Parallel development in Structured State Space Models (SSMs), such as S4 (Gu et al., 2021) and Mamba (Gu and Dao, 2023), utilized discretized continuous-time dynamics to achieve similar linear scaling. Recently, Mamba-2 (Dao and Gu, 2024) unified these paradigms under Structured State Space Duality (SSD), proving that selective SSMs are algorithmically dual to linear attention with semi-separable masks. While Mamba-2 and related SSD-style models focus on hardware utilization (via chunk-parallel scans) and expressive discretizations, they largely retain additive or gated accumulation as the state write rule. In contrast, our work re-examines the objective that induces the state update. We derive autoregressively aligned, NLMS-stabilized regression updates and sliding-window variants that remain compatible

Table 2 language model transfer to downstream tasks. Zero-shot and one-shot accuracy. Accuracies use `acc`; tasks marked with * use `acc_n` as in `lm-evaluation-harness`. Avg. is the unweighted average across the 8 tasks. Boldface marks the best recurrent model in each column.

Model	Accuracy ↑								
	PIQA	Hella.*	Wino.	ARC-e	ARC-c*	OBQA*	Social IQA	SciQ	Avg.
Zero-shot (0-shot)									
124M-130M parameters									
(124M) Transformer (w. RoPE)	65.67	37.54	51.70	52.36	27.65	31.60	38.84	79.90	48.16
(130M) RetNet/LightningAttn	64.91	35.36	49.64	57.62	26.28	32.20	37.97	80.40	48.05
(130M) Mamba-2	66.32	36.89	50.75	58.16	26.62	32.60	38.38	80.70	48.80
(130M) DeltaNet	66.38	37.15	**52.33**	57.37	26.79	**34.00**	39.00	78.00	48.88
(130M) Gated DeltaNet	65.51	**37.75**	49.72	58.88	**27.90**	31.60	38.28	80.60	48.78
Ours									
(130M) **Falcon-1A.1 (QK-ℓ_2-norm, ctxβ-ctxλ)**	66.05	37.27	50.67	57.62	27.65	31.60	38.95	81.10	48.86
(130M) **Falcon-1A.2 (QK-ℓ_2-norm, ctxη-ctxλ)**	**67.03**	37.29	**52.33**	57.37	25.94	33.20	38.84	**82.40**	**49.30**
(130M) **Falcon-1A.3 (QK-RMSNorm, ctxη-ctxλ)**	66.10	37.55	50.12	**59.01**	26.79	32.00	37.82	82.20	48.95
(130M) **Falcon-3A.3 (QK-RMSNorm, ctxη-ctxλ)**	65.34	37.30	50.99	57.37	26.37	33.60	**39.41**	81.60	49.00
One-shot (1-shot)									
(124M) Transformer (w. RoPE)	66.43	37.55	50.28	59.64	29.01	30.00	39.82	84.60	49.67
(130M) RetNet/LightningAttn	65.13	35.19	49.64	56.78	25.85	28.80	36.44	81.50	47.42
(130M) Mamba-2	66.70	36.61	51.07	58.63	26.96	**32.40**	37.97	82.90	49.16
(130M) DeltaNet	66.27	36.67	50.36	57.70	27.39	32.00	37.72	79.90	48.50
(130M) Gated DeltaNet	65.40	**37.81**	51.30	58.29	26.88	30.00	37.26	81.60	48.57
Ours									
(130M) **Falcon-1A.1 (QK-ℓ_2-norm, ctxβ-ctxλ)**	**66.81**	37.41	50.75	57.53	**27.99**	**32.40**	37.92	81.80	49.08
(130M) **Falcon-1A.2 (QK-ℓ_2-norm, ctxη-ctxλ)**	66.38	36.97	**52.96**	57.32	27.82	31.40	37.56	**83.20**	**49.20**
(130M) **Falcon-1A.3 (QK-RMSNorm, ctxη-ctxλ)**	65.78	37.22	49.72	**58.88**	27.73	31.40	37.97	82.40	48.89
(130M) **Falcon-3A.3 (QK-RMSNorm, ctxη-ctxλ)**	65.72	36.47	51.78	58.29	26.54	32.00	**38.23**	**83.20**	49.03

Table 3 Teacher-forced length generalization on variable-digit addition. Higher is better.

Model	Best step	Val. acc.	Mean acc.	Acc@d33/d48
Transformer (w. RoPE)	2000	100.0	65.8	97.0/49.0
RetNet/LightningAttn	2000	99.7	82.9	99.0/63.0
Mamba-2	2000	100.0	75.2	100.0/51.0
Ours				
Falcon-1A.1 (QK-ℓ_2-norm, ctxβ-ctxλ)	1900	100.0	80.6	100.0/59.0
Falcon-1A.2 (QK-ℓ_2-norm, ctxη-ctxλ)	2000	100.0	85.2	100.0/63.0
Falcon-1A.3 (QK-RMSNorm, ctxη-ctxλ)	1900	99.8	85.9	100.0/69.0
Falcon-3A.3 (QK-RMSNorm, ctxη-ctxλ)	2000	99.9	**87.2**	100.0/69.0

with SSD-style chunk-parallel training.

Fast Weights and Delta Networks. The concept of Fast Weights (Schmidhuber, 1992) states that a neural network can maintain a high-capacity short-term memory matrix updated by the immediate

input stream. Schlag et al. (2021) formalized this as Linear Transformers are Secretly Fast Weight Programmers, proposing the Delta Network, which updates the state via a gradient step on the prediction error rather than Hebbian addition. Yang et al. (2024a) extended this with Gated Delta Networks, introducing Mamba-style gating to improve stability. However, these prior approaches often treat the update rule as an architectural choice rather than an optimization problem.

Adaptive Filtering and Online Regression. The delta-rule updates used in fast-weight models are closely related to classical adaptive filtering, where the LMS delta rule and its normalized variant (NLMS) are standard algorithms for scale-robust online regression, and second-order methods such as recursive least squares (RLS) provide exact online ridge updates at higher cost (Sayed, 2011). Our NLMS-based Falcon-2 update can be viewed as importing these stability/normalization principles into fast-weight attention under strict causality, while remaining compatible with modern parallel implementations of linear recurrences (Yang et al., 2024b; Dao and Gu, 2024; Cirone and Salvi, 2025).

In-Context Learning as Implicit Optimization. Recent work interprets in-context learning and test-time adaptation as implicit optimization in feature space (Von Oswald et al., 2023; Ahn et al., 2023; Cheng et al., 2023; Sun et al., 2024; Wang et al., 2025). MesaNet (von Oswald et al., 2025) makes this viewpoint explicit by solving least-squares problems inside the forward pass, while Titans and ATLAS optimize internal memory objectives over the stream (Behrouz et al., 2024, 2025). Our setting is stricter: the adapted object is a bounded fast-memory state, updated online under causal next-latent alignment. This yields normalized first-order and sliding-window rules that remain compatible with chunk-parallel training.

7 Conclusion

We recast recurrent sequence modeling as online continual learning with an explicit fast-memory objective. Under read-after-write, the causal training pair is $\phi(\mathbf{k}_{t-1}) \to \mathbf{v}_t$, yielding a unified family of normalized fast-weight updates: Falcon-1/Falcon-2/Falcon-3 for regression and Falcon-1A/Falcon-2A/Falcon-3A for inner-product writes. The number denotes scalar, per-column, and sliding-window dynamics, respectively, while the suffix "A" denotes the inner-product objective. This viewpoint cleanly separates plasticity, forgetting, and bounded rehearsal while remaining compatible with chunk-parallel training. In the current empirical study, the evaluated scalar and sliding inner-product variants remain competitive in language modeling and improve arithmetic length extrapolation.

References

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 36:45614–45650, 2023.

Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. *Advances in neural information processing systems*, 29, 2016.

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

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025.

Christian Bischof and Charles Van Loan. The wy representation for products of householder matrices. *SIAM Journal on Scientific and Statistical Computing*, 8(1):s2–s13, 1987.

Xiang Cheng, Yuxin Chen, and Suvrit Sra. Transformers implement functional gradient descent to learn non-linear functions in context. *arXiv preprint arXiv:2312.06528*, 2023.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Nicola Muca Cirone and Cristopher Salvi. Parallelflow: Parallelizing linear transformers via flow discretization. *arXiv preprint arXiv:2504.00492*, 2025.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

Riccardo Grazzi, Julien Siems, Arber Zela, Jörg KH Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues. *arXiv preprint arXiv:2411.12537*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Geoffrey E Hinton and David C Plaut. Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the Cognitive Science Society*, pages 177–186, 1987.

BL Ho and Rudolf E Kálmán. Effective construction of linear state-variable models from input/output functions: Die konstruktion von linearen modeilen in der darstellung durch zustandsvariable aus den beziehungen für ein-und ausgangsgrößen. *at-Automatisierungstechnik*, 14(1-12):545–548, 1966.

Jiaxi Hu, Yongqi Pan, Jusen Du, Disen Lan, Xiaqiang Tang, Qingsong Wen, Yuxuan Liang, and Weigao Sun. Comba: Improving bilinear rnns with closed-loop control. *arXiv preprint arXiv:2506.02475*, 2025.

Łukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228*, 2015.

Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory, 1961.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems, 1960.

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

Sun-Yuan Kung. A new identification and model reduction algorithm via singular value decomposition. In *Proc. 12th asilomar conf. on circuits, systems and computer*, pages 705–714, 1978.

Sun-Yuan Kung and D Lin. A state-space formulation for optimal hankel-norm approximations. *IEEE Transactions on Automatic Control*, 26(4):942–946, 1981.

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

Zicheng Liu, Siyuan Li, Li Wang, Zedong Wang, Yunfan Liu, and Stan Z Li. Short-long convolutions help hardware-efficient linear attention to focus on long sequences. *arXiv preprint arXiv:2406.08128*, 2024b.

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. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR, 2023.

Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Lightning attention-2: A free lunch for handling unlimited sequence lengths in large language models. *arXiv preprint arXiv:2401.04658*, 2024.

Ali H Sayed. *Adaptive filters*. John Wiley & Sons, 2011.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR, 2021.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, et al. Mesanet: Sequence modeling by locally optimal test-time training. *arXiv preprint arXiv:2506.05233*, 2025.

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

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

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

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *arXiv preprint arXiv:2406.06484*, 2024b.

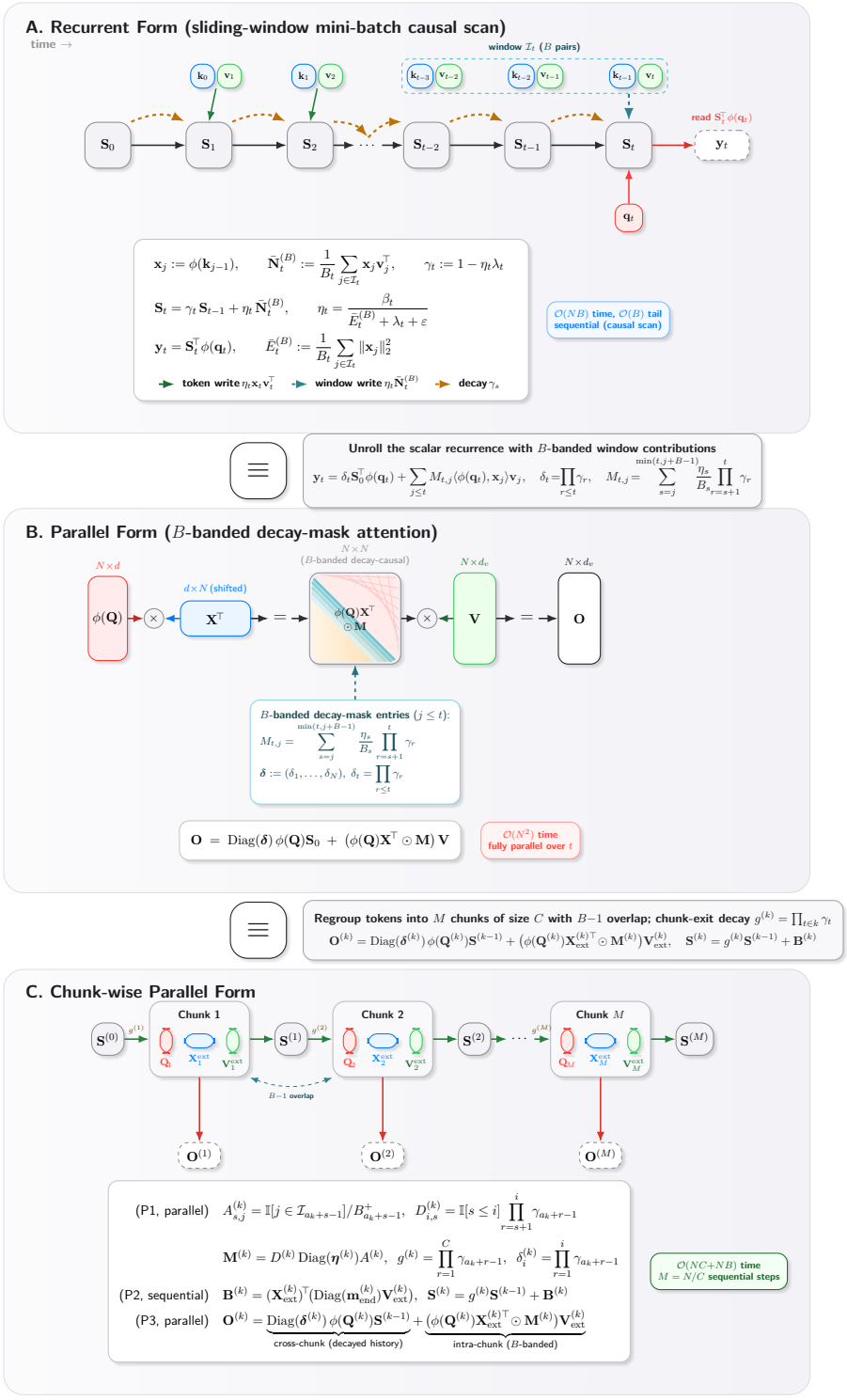

Figure 10 Three equivalent views of Falcon-3A. (A) The recurrent form maintains a bounded matrix state $\mathbf{S}_t \in \mathbb{R}^{d_x \times d_v}$. (B) Unrolling the scalar recurrence with the windowed write yields a masked-attention form with the same shifted features but a *B-banded decay-weighted* causal mask. (C) The chunk-wise form splits the sequence into M chunks of size C.

Appendix

A More on Background

Notation. We reuse the main-text regression notation: $\mathbf{x}_t$ is the write feature, typically $\phi(\mathbf{k}_{t-1})$ under next-latent alignment; $\mathbf{y}_t$ is the target, typically $\mathbf{v}_t$; and $\mathbf{S}_t$ is the fast-weight state. The residual is $\mathbf{r}_t := \mathbf{y}_t - \mathbf{S}_{t-1}^\top \mathbf{x}_t$.

A.1 Recursive Least Squares

Recursive least squares (RLS) realizes the exact cumulative ridge-regression solution online via the matrix inversion lemma. We record the constant-ridge, zero-prior-mean case $\lambda > 0$ with $\mathbf{S}_0 = \mathbf{0}$. This is a reference solver, not the exact closed-form solution for the time-varying λ_t recurrences used elsewhere in the paper. Let

$$\mathbf{P}_{t-1} := \big(\lambda \mathbf{I} + \textstyle\sum_{s=1}^{t-1} \mathbf{x}_s \mathbf{x}_s^\top\big)^{-1}, \qquad \mathbf{P}_0 := \lambda^{-1}\mathbf{I} \;\; (\lambda > 0).$$

Define the gain and covariance updates

$$\mathbf{g}_t = \frac{\mathbf{P}_{t-1}\mathbf{x}_t}{1 + \mathbf{x}_t^\top \mathbf{P}_{t-1}\mathbf{x}_t}, \qquad \mathbf{P}_t = \mathbf{P}_{t-1} - \mathbf{g}_t \mathbf{x}_t^\top \mathbf{P}_{t-1}. \tag{A.1}$$

Then the parameter update is

$$\mathbf{r}_t := \mathbf{y}_t - \mathbf{S}_{t-1}^\top \mathbf{x}_t, \qquad \mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{g}_t\, \mathbf{r}_t^\top. \tag{A.2}$$

These recursions cost $O(d_x^2 + d_x d_v)$ per step, where $d_x := \dim(\mathbf{x}_t)$, and satisfy that $\mathbf{S}_t$ is the exact ridge-regression solution after processing t samples, i.e. the minimizer of the cumulative ridge objective on $\{(\mathbf{x}_s, \mathbf{y}_s)\}_{s=1}^t$ with zero prior mean. A nonzero initial state corresponds instead to a prior-centered ridge objective. Exact online solvers of this form are much harder to parallelize than first-order recurrences, which is why MesaNet (von Oswald et al., 2025) uses an approximate chunk-parallel approach.

A.2 First-Order Online Ridge

Algorithm 5 records the reference sequential first-order update used throughout the main text.

A.3 SSM Discretization

Over step size Δ_t, the exact zero-order-hold discretization of the continuous-time linear system is

$$\mathbf{h}_t = \bar{\mathbf{A}}_t \mathbf{h}_{t-1} + \bar{\mathbf{B}}_t x_t, \qquad \bar{\mathbf{A}}_t = e^{\Delta_t \mathbf{A}_t}, \qquad \bar{\mathbf{B}}_t = \int_0^{\Delta_t} e^{(\Delta_t - s)\mathbf{A}_t} \mathbf{B}_t \, ds. \tag{A.3}$$

When $\mathbf{A}_t$ is diagonal, the input integral admits the elementwise closed form

$$\bar{\mathbf{B}}_t = \Big(\mathbf{A}_t^{-1}(e^{\Delta_t \mathbf{A}_t} - \mathbf{I})\Big)\, \mathbf{B}_t,$$

where the factor $\mathbf{A}_t^{-1}(e^{\Delta_t \mathbf{A}_t} - \mathbf{I})$ is interpreted elementwise and uses the limit $(e^{\Delta a} - 1)/a \to \Delta$ as $a \to 0$. A first-order approximation is

$$\bar{\mathbf{B}}_t \approx \Delta_t \mathbf{B}_t, \qquad \mathbf{h}_t \approx e^{\Delta_t \mathbf{A}_t} \mathbf{h}_{t-1} + \Delta_t \mathbf{B}_t x_t,$$

valid when $\|\Delta_t \mathbf{A}_t\|$ is small.

Algorithm 5 Online ridge via SGD with batch size 1

Require: Keys $\{\mathbf{k}_t\}_{t=1}^T$, feature map ϕ (default identity), values $\{\mathbf{v}_t\}_{t=1}^T$, boundary write feature $\mathbf{x}_1 := \mathbf{0}$ and $\mathbf{x}_t := \phi(\mathbf{k}_{t-1})$ for $t \geq 2$, ridge coefficients $\{\lambda_t\}_{t=1}^T$ with $\lambda_t \geq 0$, gain $\beta_t \in (0, 2)$, stabilizer $\varepsilon \geq 0$ (set η_t:=0 if the denominator is 0), init $\mathbf{S}_0 = \mathbf{0}$.

for $t = 1, \ldots, T$ **do**
  $\mathbf{x}_t \leftarrow \mathbf{0}$ if $t = 1$, else $\phi(\mathbf{k}_{t-1})$ ▷ Write feature
  $\mathbf{y}_t \leftarrow \mathbf{v}_t$ ▷ Target
  $\widehat{\mathbf{y}}_t \leftarrow \mathbf{S}_{t-1}^\top \mathbf{x}_t$ ▷ Fast-memory prediction
  $\mathbf{r}_t \leftarrow \mathbf{y}_t - \widehat{\mathbf{y}}_t$ ▷ Residual
  $\eta_t \leftarrow 0$ if $t = 1$ or $\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon = 0$, else $\beta_t / (\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon)$
  $\mathbf{S}_t \leftarrow (1 - \eta_t \lambda_t)\mathbf{S}_{t-1} + \eta_t \, \mathbf{x}_t \mathbf{r}_t^\top$
end for

B Experimental Setup

B.1 Language Model Training Setup

For the language model runs in Section 5.1, all models are trained in bfloat16 with AdamW, tied input/output embeddings, Pre-Norm RMSNorm, no bias terms, and no dropout. We use μP-style width scaling, base learning rate 10^{-3} with cosine decay, 2,000 warmup steps, $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 0.1, and gradient clipping at 1.0. Each run is performed on a single 4-GPU node with NVIDIA H100 or H200 GPUs.

C Implementation Details

C.1 Scaled Fast-Weight Recurrences

The normalized counterpart of the denominator-free scaled linear-attention recurrence in Section 4.1 is

$$\mathbf{y}_t = \frac{\mathbf{S}_t^\top \phi(\mathbf{q}_t)}{\mathbf{z}_t^\top \phi(\mathbf{q}_t) + \varepsilon_{\text{attn}}}, \qquad \mathbf{S}_t = \gamma_t \mathbf{S}_{t-1} + \eta_t \, \mathbf{x}_t \mathbf{v}_t^\top, \qquad \mathbf{z}_t = \gamma_t \mathbf{z}_{t-1} + \eta_t \, \mathbf{x}_t,$$

$$\alpha_t := \min(\eta_t \lambda_t, 1 - \varepsilon_\gamma), \qquad \gamma_t := 1 - \alpha_t.$$

When the clamp is inactive, this is exactly the update with carry $1 - \eta_t \lambda_t$. If the clamp activates, the shrinkage path is instead the positive-decay surrogate with effective coefficient α_t / η_t for $\eta_t > 0$. With $\mathbf{z}_0 \geq 0$ and elementwise nonnegative feature maps, this normalized form remains attention-like only when $\gamma_t \geq 0$ for all t; the clamp enforces $\gamma_t \geq \varepsilon_\gamma > 0$. The additive normalized case in Eq. (2.3) is recovered by $\eta_t \equiv 1$ and $\lambda_t \equiv 0$.

For scaled non-sliding variants, let

$$\bar{\lambda}_t := \lambda_{\text{scale}} \, \sigma(\widetilde{\lambda}_t), \qquad \lambda_t^{\text{eff}} := \bar{\lambda}_t \, \|\mathbf{x}_t\|_2^2.$$

Then

$$\eta_t = \frac{\beta_t}{\|\mathbf{x}_t\|_2^2 + \lambda_t^{\text{eff}} + \varepsilon}, \qquad \gamma_t = 1 - \eta_t \lambda_t^{\text{eff}}.$$

When $\varepsilon = 0$ and $\|\mathbf{x}_t\|_2 > 0$, both the scalar decay fraction and the normalized projector term are invariant to uniform rescaling of $\mathbf{x}_t$. Sliding regression variants replace $\|\mathbf{x}_t\|_2^2$ with $\mu_t^{(B)} := \lambda_{\max}(\bar{\mathbf{C}}_t^{(B)})$; sliding inner-product variants use the write-energy statistic $\bar{E}_t^{(B)}$.

C.2 Signed-Feature Normalizers

The normalized linear-attention read

$$\mathbf{y}_t = \frac{\mathbf{S}_t^\top \phi(\mathbf{q}_t)}{\mathbf{z}_t^\top \phi(\mathbf{q}_t) + \varepsilon_{\text{attn}}}$$

is mathematically safest when the read denominator is nonnegative, as with elementwise nonnegative feature maps satisfying $\phi(\cdot) \geq 0$ and $\mathbf{z}_t \geq 0$. In signed-feature settings, adding a positive constant does not by itself prevent the denominator from vanishing or changing sign:

$$\mathbf{z}_t^\top \phi(\mathbf{q}_t) + \varepsilon_{\text{attn}}$$

can still be zero or negative. Thus denominator-free inner-product reads are the default signed-feature interpretation in this paper. If a normalized signed-feature read is desired, the implementation must use an explicit safe denominator, clamp, or other sign-stable normalization.

C.3 Step Size and Gating Conventions

Prior DeltaNet literature often writes the raw gradient step size as β_t. We reserve η_t for the actual step size and use β_t for the dimensionless NLMS gain. This convention separates the learned plasticity control from the normalization statistic:

$$\eta_t = \frac{\beta_t}{\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon}$$

for the rank-one regression rule, and analogously for the windowed and inner-product variants.

The standard Delta update

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

contains only implicit forgetting through the rank-one edit. Gated Delta Networks (Yang et al., 2024a) add an explicit scalar carry:

$$\mathbf{S}_t = g_t(\mathbf{I} - \eta_t \mathbf{k}_t \mathbf{k}_t^\top)\mathbf{S}_{t-1} + \eta_t \mathbf{k}_t \mathbf{v}_t^\top, \qquad g_t \in [0, 1]. \tag{C.1}$$

We write this gate as g_t to avoid overloading the Falcon notation, where $\alpha_t := \eta_t \lambda_t$ is a derived decay fraction and $\gamma_t := 1 - \alpha_t$ is the carry. The gated Delta rule and Falcon-style ridge shrinkage both provide global state decay; the main distinction is that Falcon ties the carry to the local objective and the normalized step-size parameterization.

C.4 Relation to Test-Time Training and Titans

Test-Time Training (TTT) updates a subset of model parameters at inference by taking gradient steps on an internal, self-supervised objective, and recent work formalizes this mechanism as *test-time regression* in a feature space (Sun et al., 2024; Wang et al., 2025). Our fast-weight update is a constrained, single-step instance of this paradigm: the recurrent state $\mathbf{S}$ is the adapted parameter and Eq. (3.1) is the inner objective. Relative to generic TTT, we enforce strict autoregressive causality by updating only after $\mathbf{v}_t$ is revealed and by writing it under the prefix write feature available at prediction time, $\mathbf{x}_t = \phi(\mathbf{k}_{t-1})$. Titans and ATLAS similarly view the recurrent state as fast memory optimized online with an internal objective over the stream (Behrouz et al., 2024, 2025); Falcon-3 can be read as a sliding-window specialization of that view. Our contribution is to make the write-feature/target alignment explicit and to derive normalized first-order rules and sliding-window variants compatible with SSD-style chunk-parallel training.

C.5 Timing and Boundary Conventions

We use the read-after-write (RAW) indexing convention throughout the paper: after token t is observed and written, the updated state $\mathbf{S}_t$ is read to predict token $t{+}1$. This is the standard Transformer indexing shifted by one step relative to the read-before-write (RBW) view that predicts token t from prefix $1{:}t{-}1$. Under next-latent alignment, the RAW update therefore uses the causal pair $(\phi(\mathbf{k}_{t-1}), \mathbf{v}_t)$, or equivalently $(\phi(\mathbf{k}_i), \mathbf{v}_{i+1})$ under standard indexing. Figure 11 visualizes the RAW/RBW timing conventions used throughout the paper.

The four cells in Fig. 11 correspond to different local-objective conventions. Under RAW, the shifted pairing $(\phi(\mathbf{k}_{t-1}), \mathbf{v}_t)$ is aligned with the prefix feature that was available when $\mathbf{v}_t$ was predicted. The unshifted RAW pairing $(\phi(\mathbf{k}_t), \mathbf{v}_t)$ is still causal for next-token prediction, but it optimizes a different local objective based on same-step features. Likewise, the shifted RBW pairing is internally consistent, but it updates memory after the read that produced the position-t prediction.

We initialize $\mathbf{S}_0 = \mathbf{0}$ and impose the feature-space boundary condition $\mathbf{x}_1 := \mathbf{0}$. In the identity-feature case this can be realized by a zero raw BOS key; for a generic feature map the condition is imposed directly in feature space. Since $\mathbf{x}_1 = \mathbf{0}$ suppresses only the data write, not necessarily ridge shrinkage, we also set $\eta_1 := 0$ at this boundary sentinel. Equivalently, $(\alpha_1, \gamma_1) = (0, 1)$ in the derived-variable notation used for log-space unrolling. This convention is essential when a nonzero state is carried across segments.

With this convention, the internal fast-memory prediction at step t is $\widehat{\mathbf{y}}_t := \mathbf{S}_{t-1}^\top \mathbf{x}_t$, whereas the model readout at position t uses the updated state $\mathbf{S}_t$. The two quantities should not be conflated.

C.6 Implementation Notes and Related Update Rules

Gain and step size. We use $\beta_t \in (0, 2)$ for the dimensionless NLMS gain and η_t for the resulting normalized step size. Some prior DeltaNet papers use β_t for the raw step size; we keep the two quantities distinct. In the unclamped $\varepsilon = 0$ analysis, $\beta_t > 1$ induces a negative eigenvalue along the current write-feature direction, i.e., a stable sign flip that can improve state tracking (Grazzi et al., 2024).

Ridge as decay. For $\lambda_t > 0$, the ridge term contributes the scalar carry $\gamma_t := 1 - \eta_t \lambda_t$ in Eq. (3.3). The full linear transition is $\mathbf{A}_t = \gamma_t \mathbf{I}_{d_x} - \eta_t \mathbf{x}_t \mathbf{x}_t^\top$, so the subspace orthogonal to $\mathbf{x}_t$ sees eigenvalue γ_t,

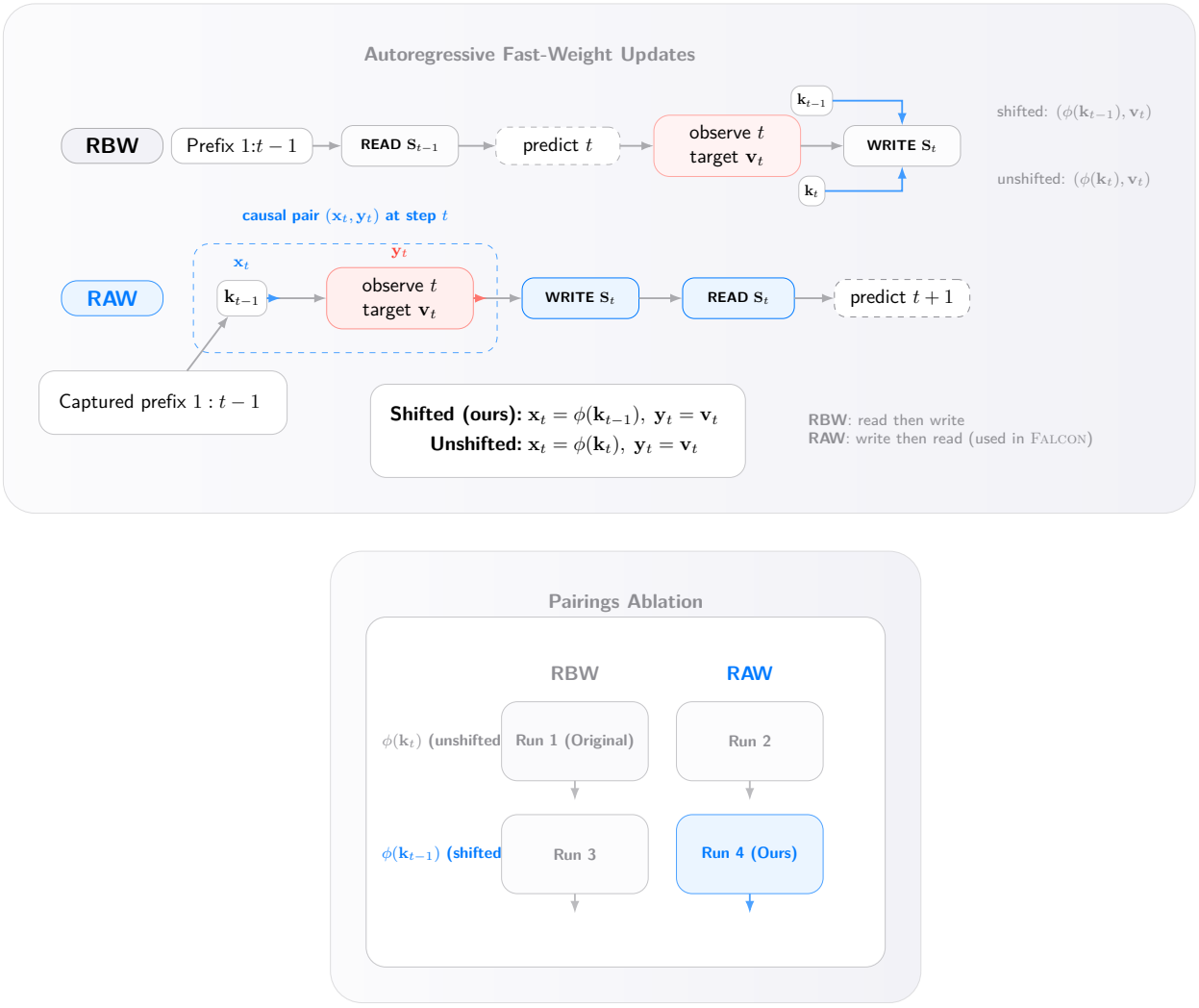

Figure 11 Formal timing conventions. Above: We explicitly distinguish *read-before-write* (RBW) and *read-after-write* (RAW) autoregressive fast-weight updates. Under the RAW convention used in FALCON, the causally aligned training pair at step t is $(\phi(\mathbf{k}_{t-1}), \mathbf{v}_t)$, i.e., a prefix write feature paired with the newly revealed target. Below: A 2×2 design over timing convention $\{\text{RBW}, \text{RAW}\}$ and write pairing {same-step, shifted} separates causal timing from mere index shifting; in the kernelized setting these pairings are understood in write-feature space, i.e. $\{\phi(\mathbf{k}_t), \phi(\mathbf{k}_{t-1})\}$.

while the write-feature direction sees eigenvalue $1 - \eta_t(\|\mathbf{x}_t\|_2^2 + \lambda_t)$. With $\varepsilon = 0$ and Eq. (3.4), this directional eigenvalue equals $1 - \beta_t$. When log-space unrolling requires positive decay, we clamp the derived fraction $\alpha_t := \eta_t \lambda_t$; if the clamp activates, the implemented shrinkage may be interpreted as using an effective coefficient $\widetilde{\lambda}_t := \alpha_t / \eta_t \le \lambda_t$ whenever $\eta_t > 0$ (and 0 when $\eta_t = 0$).

Log-space chunk-local renormalization. In long-context settings, direct products of positive decay factors can become numerically fragile in mixed precision. For scalar-decay recurrences of the form

$$\mathbf{S}_t = (\gamma_t \mathbf{I} - \eta_t \mathbf{x}_t \mathbf{x}_t^\top)\mathbf{S}_{t-1} \; + \; \eta_t \, \mathbf{x}_t \mathbf{y}_t^\top, \qquad \gamma_t > 0,$$

with $\mathbf{y}_t$ the write target (e.g. $\mathbf{v}_t$), the unclamped carry is $\gamma_t = 1 - \eta_t \lambda_t$. Whenever positivity is not guaranteed a priori, we instead define

$$\alpha_t := \eta_t \lambda_t, \qquad \alpha_t \leftarrow \min(\alpha_t, 1 - \varepsilon_\gamma), \qquad \gamma_t := 1 - \alpha_t,$$

and compute

$$\log \gamma_t := \mathrm{log1p}(-\alpha_t)$$

in fp32. Let

$$c_0 := 1, \qquad c_t := \prod_{r=1}^{t} \gamma_r, \qquad \widetilde{\mathbf{S}}_t := \mathbf{S}_t / c_t.$$

Then

$$\widetilde{\mathbf{S}}_t = (\mathbf{I} - \widetilde{\eta}_t \, \mathbf{x}_t \mathbf{x}_t^\top)\widetilde{\mathbf{S}}_{t-1} \; + \; \widetilde{\eta}_t \, \mathbf{x}_t \widetilde{\mathbf{y}}_t^\top, \qquad \widetilde{\eta}_t := \frac{\eta_t}{\gamma_t}, \qquad \widetilde{\mathbf{y}}_t := \frac{\mathbf{y}_t}{c_{t-1}}.$$

Hence exact equivalence requires *both* step-size rescaling and inverse-decay rescaling of the write target; rescaling η_t alone is not sufficient. In chunk-parallel implementations, we do not form global products c_t; instead, we accumulate $\log \gamma_t$ within each chunk and apply the equivalent local renormalization to boundary states and write targets. For normalized readouts of the form

$$\mathbf{y}_t = \frac{\mathbf{S}_t^\top \phi(\mathbf{q}_t)}{\mathbf{z}_t^\top \phi(\mathbf{q}_t) + \varepsilon_{\mathrm{attn}}},$$

exact output preservation additionally requires forming the ratio after undoing the common rescaling, or equivalently, rescaling the stabilizer by the same factor. With a fixed $\varepsilon_{\mathrm{attn}} > 0$ and directly using the renormalized states inside the ratio, the equivalence is only approximate. The denominator-free recurrence is unaffected by this caveat.

Optional local convolution. In some implementations, we apply a short causal convolution before forming $(\mathbf{q}, \mathbf{k}, \mathbf{v})$, following prior observations that local mixing can improve associative recall (Fu et al., 2022; Liu et al., 2024b). This choice is orthogonal to the fast-memory update itself and is not required by the theory.

Identity-matrix notation. We write $\mathbf{I}_{d_x}$ for the $d_x \times d_x$ identity in feature space, where $d_x \triangleq \dim(\mathbf{x}_t)$ (so $d_x = d$ for raw keys and $d_x = m$ for kernelized write-features), and $\mathbf{I}_{d_v}$ for the $d_v \times d_v$ identity in value space. When the dimension is unambiguous, we write $\mathbf{I}$.

Scale-coupled ridge. Appendix C.1 gives the exact non-sliding scale-invariance identities. Sliding regression uses the smoothness statistic $\mu_t^{(B)} := \lambda_{\max}(\bar{\mathbf{C}}_t^{(B)})$, while sliding inner-product writes use the window energy $\bar{E}_t^{(B)}$. In the current sliding-state implementation, this multiplier is treated as statistics-only (detached / stop-gradient) when forming the actual shrinkage coefficient; the step-size denominator still uses the live statistic.

Small-Gram smoothness for Falcon-3. For the sliding regression rule, the denominator uses

$$\mu_t^{(B)} = \lambda_{\max}(\bar{\mathbf{C}}_t^{(B)}) = \frac{\lambda_{\max}(\mathbf{X}_t^\top \mathbf{X}_t)}{B_t},$$

where $\mathbf{X}_t \in \mathbb{R}^{d_x \times B_t}$ stacks the active-window write-features. Because $B_t \le B$ is small in the intended regime, one can either diagonalize the $B_t \times B_t$ Gram matrix exactly or approximate its top eigenvalue with a few power iterations; the theory in the main text is stated with the exact quantity.

Relation to RWKV-7 and Kimi Linear Attention. RWKV-style recurrent LLMs (Peng et al., 2023) and more recent delta-rule variants (e.g., RWKV-7 and Kimi Linear Attention (Team et al., 2025)) enrich Delta-style fast weights with learned gating and in-context learning-rate control. Our formulation is complementary: we ground the write rule in a causal next-latent regression objective, make the required one-step key shift explicit, and stabilize learning with NLMS normalization. The same normalization/alignment can be used as a drop-in replacement for the local update inside delta-style blocks, independent of their particular gating or mixing parameterization.

C.7 Sliding-Window Boundary State

For Falcon-3 and Falcon-3A, the matrix state $\mathbf{S}_t$ alone is not Markov for exact continuation. Exact continuation across a segment boundary also requires a fixed-width tail containing the last $B-1$ causal pairs $\{(\mathbf{x}_j, \mathbf{v}_j)\}_{j=\max(2,t-B+2)}^{t}$, or an equivalent rolling-window representation. Resetting this tail changes the near-boundary update and should be treated as an explicit boundary reset.

For Falcon-3, Algorithm 2 is written for a fresh sequence with $\mathbf{x}_1 = \mathbf{0}$. Algorithm 3 realizes the same overlap offline by gathering the extended write-pair set $J = \{p, \dots, b\}$ at the chunk entry. Passing only the matrix boundary state between disjoint segments is therefore not exact.

For Falcon-3A, aggregated statistics such as $(\bar{\mathbf{N}}_t^{(B)}, \bar{E}_t^{(B)}, B_t)$ are not sufficient for indefinite continuation, because the next step must remove the oldest contribution exactly. The masked-attention unrolling in Section 4.6 assumes the same overlap is available.

Away from boundaries, when $\lambda_t = 0$ and $\eta_t = \eta$ is constant, each token appears in exactly B consecutive window-averages with weight $1/B$, so its cumulative injection coefficient is $\sum_{s=j}^{j+B-1} \eta/B = \eta$, independent of B. This does not make the memory strictly local: after those B direct injections, the token's effect can persist through later carry factors. The window therefore, controls an *effective* memory horizon rather than strict local support; strict locality would require explicit truncation or a hard zero-carry override.

D Falcon-3A Mask and Backward Pass

D.1 Mask Identities and Chunk-Local Evaluation

This subsection records the mask formulas underlying Section 4.6. Starting from

$$\mathbf{O} = \mathrm{Diag}(\boldsymbol{\delta})\,\mathbf{Q}\mathbf{S}_0 \;+\; (\mathbf{Q}\mathbf{X}^\top \odot \mathbf{M})\mathbf{V}, \qquad \boldsymbol{\delta} := (\delta_1, \ldots, \delta_L)^\top,$$

the fast-weight contribution is a masked-attention term and the boundary contribution is a decayed-history term. The causal mask is step dependent (and input-conditioned through η_s when the write energy varies): each coefficient $M_{t,j}$ is a sum of at most B decayed write coefficients η_s/B_s, and it is generally nonzero for all $j \le t$. Moreover, $M_{t,j}$ need not be monotone in the lag $t-j$ because token j is re-injected into $\mathbf{N}_s^{(B)}$ for $s = j, \ldots, j+B-1$ before subsequent decay acts on the state.

In the stationary case $\eta_s = \eta$ and $\gamma_s = \gamma \in (0,1)$,

$$M_{t,j} = \frac{\eta}{B} \sum_{u=\max(0,\,t-j-B+1)}^{t-j} \gamma^u,$$

for full-window interior positions where $B_s = B$ throughout the contributing range. Near sequence or segment boundaries, replace B by the realized B_s in the sum. This makes the moving-sum then exponential-tail structure explicit. All expressions here follow the paper-wide read-after-write convention: the readout at position t uses the updated state $\mathbf{S}_t$ and is used to predict token $t{+}1$.

Mask structure. The mask $M_{t,j}$ can be written explicitly as

$$M_{t,j} = \sum_{s=j}^{\min(t,j+B-1)} \frac{\eta_s}{B_s} \prod_{r=s+1}^{t} (1 - \eta_r \lambda_r), \tag{D.1}$$

with the implemented recurrence replacing $1 - \eta_r\lambda_r$ by the clamped γ_r when needed. This separates the window-average accumulation from the global multiplicative carry.

Log-space cumulative decay. Directly forming $\prod_{r=s+1}^{t} \gamma_r$ can underflow over long sequences, especially in reduced precision. Define the log cumulative decay

$$\alpha_t := \min(\eta_t\lambda_t,\, 1-\varepsilon_\gamma), \qquad \zeta_0 := 0, \qquad \zeta_t := \zeta_{t-1} + \log\gamma_t \quad (t \ge 1),$$

with $\log\gamma_t = \mathrm{log1p}(-\alpha_t)$ computed in fp32, so that $\prod_{r=s+1}^{t}\gamma_r = \exp(\zeta_t - \zeta_s)$ and

$$M_{t,j} = \sum_{s=j}^{\min(t,j+B-1)} \frac{\eta_s}{B_s} \exp(\zeta_t - \zeta_s). \tag{D.2}$$

Chunk-local renormalization. Even with the log representation, explicitly materializing global ζ_t and then forming $\exp(\zeta_t - \zeta_s)$ at very large lags can underflow in reduced precision and is unnecessary for chunked kernels. All chunk-parallel algorithms in this paper, therefore, reset the log-prefix to 0 at chunk boundaries, use only within-chunk differences $u_i - u_s$ with $u_0 = 0$, and propagate boundary states via the chunk-exit decay $g^{(k)} = \prod_{t\in\text{chunk }k}\gamma_t$. This is exactly the local log-decay trick used by Algorithm 4.

Chunk-parallel evaluation. For implementation, it is helpful to expose the mask as the composition of (i) a causal decay kernel and (ii) a B-banded moving-average window operator:

$$D_{t,s} := \mathbb{I}[s \le t] \prod_{r=s+1}^{t} \gamma_r, \qquad B_s^+ := \max(1, B_s), \qquad A_{s,j} := \frac{\mathbb{I}[j \in \mathcal{I}_s]}{B_s^+}, \qquad \mathbf{M} = \mathbf{D}\ \mathrm{Diag}(\boldsymbol{\eta})\, \mathbf{A},$$

where $\boldsymbol{\eta} := (\eta_1, \dots, \eta_L)^\top$. Under our boundary convention ($\mathcal{I}_1 = \emptyset$ and $\eta_1 = 0$), the $s = 1$ row of $\mathbf{A}$ is all zeros, and for all $s \ge 2$ we have $B_s^+ = B_s$, so this agrees with $A_{s,j} = \mathbb{I}[j \in \mathcal{I}_s]/B_s$ while avoiding a $0/0$ definition at $s = 1$. This yields the three-phase chunked algorithm used in the main text: per-chunk precomputation (parallel), inter-chunk boundary propagation (short recurrence/scan), and final output materialization (parallel).

D.2 Backward Pass

The backward pass is reverse-mode differentiation through Algorithm 4. In the final (β, λ, η) parameterization, the only nontrivial scalar chain rules come from

$$\eta_t = \frac{\beta_t}{d_t}, \qquad d_t := \bar{E}_t^{(B)} + \lambda_t + \varepsilon, \qquad \alpha_t = \min(\eta_t \lambda_t, 1 - \varepsilon_\gamma), \qquad \bar{E}_t^{(B)} = \frac{1}{B_t} \sum_{j \in \mathcal{I}_t} \|\mathbf{x}_j\|_2^2.$$

Let $\bar{\eta}_t$ and $\bar{\alpha}_t$ denote the adjoints arriving from the structured-mask and log-decay paths, respectively. For unclamped steps ($\eta_t \lambda_t < 1 - \varepsilon_\gamma$),

$$\bar{\eta}_t \mathrel{+}= \lambda_t\, \bar{\alpha}_t, \qquad \bar{\lambda}_t \mathrel{+}= \eta_t\, \bar{\alpha}_t.$$

Then

$$\bar{\beta}_t \mathrel{+}= \frac{\bar{\eta}_t}{d_t}, \qquad \bar{d}_t = -\frac{\beta_t}{d_t^2}\, \bar{\eta}_t, \qquad \bar{\lambda}_t \mathrel{+}= \bar{d}_t, \qquad g_t^{(E)} \mathrel{+}= \bar{d}_t,$$

where $g_t^{(E)}$ denotes the adjoint of $\bar{E}_t^{(B)}$.

Finally, the window-energy path contributes

$$\bar{\mathbf{x}}_j \mathrel{+}= 2 \left(\sum_{t:\, j \in \mathcal{I}_t} \frac{g_t^{(E)}}{B_t} \right) \mathbf{x}_j.$$

Equivalently, in the chunked notation of Algorithm 4,

$$\bar{\mathbf{X}}_{\mathrm{ext}}^{(k)} \mathrel{+}= 2\big((\mathbf{A}^{(k)\top} \mathbf{g}^{(k,E)}) \mathbf{1}_{d_x}^\top\big) \odot \mathbf{X}_{\mathrm{ext}}^{(k)},$$

where $\mathbf{g}^{(k,E)} \in \mathbb{R}^C$ stores the adjoints of the chunk-local window energies.

If the same scale-coupled base-ridge parameterization as in FALCON-3 is used, first run the backward pass above with respect to the actual coefficients λ_t. Then apply the outer chain rule through

$$\lambda_t = \bar{\lambda}_t\, \bar{E}_t^{(B)}.$$

Thus, the base-ridge gradient is the gradient with respect to the actual coefficient multiplied by $\bar{E}_t^{(B)}$; if the multiplier is live, add the corresponding $\bar{\lambda}_t$-scaled term to the energy gradient. With

the statistics-only / stop-gradient multiplier used in the current sliding-state implementation, only the base-ridge path remains, and $\bar{E}_t^{(B)}$ is treated as detached.

When the clamp is active, multiply the $\bar{\alpha}_t$ path by the indicator $\mathbb{I}[\eta_t \lambda_t < 1 - \varepsilon_\gamma]$. The remaining derivatives through the block-causal mask, local log-prefix decays, boundary propagation, and output materialization are unchanged once η_t and γ_t have been materialized. Because the boundary step is hard-set to $\eta_1 = 0$ and $\gamma_1 = 1$, it carries no gradient to (β_1, λ_1).

E DeltaNet WY Parallelization

The basic Delta Network update in Section 3 has the form

$$\mathbf{S}_t = (\mathbf{I} - \eta_t \mathbf{x}_t \mathbf{x}_t^\top)\mathbf{S}_{t-1} + \eta_t \mathbf{x}_t \mathbf{v}_t^\top, \tag{E.1}$$

where $\mathbf{x}_t := \phi(\mathbf{k}_{t-1})$ is the shifted write feature. This corresponds to the projector-only case $\lambda_t = 0$ of Eq. (3.3). When ridge regularization / weight decay is enabled, Eq. (3.3) changes the transition to $\mathbf{A}_t = \gamma_t \mathbf{I} - \eta_t \mathbf{x}_t \mathbf{x}_t^\top$ with $\gamma_t := 1 - \eta_t \lambda_t$. For $\gamma_t > 0$ one can factor $\mathbf{A}_t = \gamma_t(\mathbf{I} - \mathbf{u}'_t \mathbf{u}'^\top_t)$ where $\mathbf{u}'_t := \sqrt{\eta_t/\gamma_t}\, \mathbf{x}_t$, and absorb $\prod \gamma_t$ into a cumulative rescaling of boundary states and bias terms. For clarity, the WY derivation below presents the projector-only case $\lambda_t = 0$.

Explicit reduction to the projector-only case (for $\lambda_t > 0$). Because γ_t is a *scalar*, one can reuse the WY machinery unchanged via a simple rescaling. Let $c_0 := 1$ and $c_t := \prod_{r=1}^t \gamma_r$ and define $\widetilde{\mathbf{S}}_t := \mathbf{S}_t / c_t$, $\widetilde{\eta}_t := \eta_t / \gamma_t$, and $\widetilde{\mathbf{v}}_t := \mathbf{v}_t / c_{t-1}$. Then the decayed recurrence is equivalent to the projector-only recurrence

$$\widetilde{\mathbf{S}}_t = (\mathbf{I} - \widetilde{\eta}_t\, \mathbf{x}_t \mathbf{x}_t^\top)\widetilde{\mathbf{S}}_{t-1} + \widetilde{\eta}_t\, \mathbf{x}_t \widetilde{\mathbf{v}}_t^\top, \qquad \mathbf{o}_t = c_t \widetilde{\mathbf{o}}_t.$$

Chunk-local renormalization. Forming c_t (or c_{t-1}^{-1}) over long sequences can underflow/overflow, all chunk-wise kernels therefore apply this rescaling *within each chunk* by resetting the log-prefix decay to 0 at chunk boundaries (see Algorithm 1 and Algorithm 8).

Step-size convention. Throughout this appendix, η_t denotes the actual step size used in the rank-one update (e.g., the NLMS step size from Eq. (3.4)). In the *projector-only* setting treated here ($\lambda_t = 0$), a gain $\beta_t \in (0, 2)$ corresponds to

$$\eta_t = \frac{\beta_t}{\|\mathbf{x}_t\|_2^2 + \varepsilon},$$

with the usual convention $\eta_t := 0$ when the denominator vanishes. Hence $\eta_t \geq 0$ and $\sqrt{\eta_t}$ is well-defined.

E.1 Affine WY Form

We explicitly define the affine structure. Write

$$\mathbf{A}_t := \mathbf{I} - \eta_t \mathbf{x}_t \mathbf{x}_t^\top \in \mathbb{R}^{d_x \times d_x}, \qquad \mathbf{B}_t := \eta_t \mathbf{x}_t \mathbf{v}_t^\top \in \mathbb{R}^{d_x \times d_v}, \tag{E.2}$$

so that the recurrence becomes $\mathbf{S}_t = \mathbf{A}_t \mathbf{S}_{t-1} + \mathbf{B}_t$ (projector-only; $\lambda_t = 0$). Eq. (E.2) reveals a rank-one structure $\mathbf{A}_t = \mathbf{I} - \mathbf{u}_t \mathbf{u}_t^\top$ where $\mathbf{u}_t := \sqrt{\eta_t}\, \mathbf{x}_t$. Bischof and Van Loan (1987) show that the

product of such matrices over a block can be written as a single WY transform (or UT transform). For a chunk of size C, the accumulated transition matrix $\widehat{\mathbf{A}} = \mathbf{A}_{t+C} \cdots \mathbf{A}_{t+1}$ is:

$$\widehat{\mathbf{A}} = \mathbf{I} - \mathbf{U}\mathbf{T}\mathbf{U}^\top, \tag{E.3}$$

where $\mathbf{U} \in \mathbb{R}^{d_x \times C}$ collects the update vectors $\mathbf{u}$, and $\mathbf{T} \in \mathbb{R}^{C \times C}$ is a triangular mixing matrix.

E.2 Chunk-Parallel Forward Pass

Similar to Yang et al. (2024b), we have the following three-phase parallel algorithm. Phase 1 computes the local operators in parallel. Phase 2 performs a fast recurrence over chunk boundaries. Phase 3 materializes the outputs by combining the propagated state with a local causal attention mechanism.

Algorithm 6 Parallel DeltaNet Training (Forward, $\lambda_t = 0$)

Require: Shifted write features $\mathbf{x} \in \mathbb{R}^{L \times d_x}$ (with $\mathbf{x}_t := \phi(\mathbf{k}_{t-1})$), queries $\mathbf{Q} \in \mathbb{R}^{L \times d_x}$, values $\mathbf{V} \in \mathbb{R}^{L \times d_v}$, step sizes $\boldsymbol{\eta} \in \mathbb{R}^L$ with $\eta_t \geq 0$ (so $\sqrt{\eta_t}$ is defined), initial state $\mathbf{S}_{\text{init}} \in \mathbb{R}^{d_x \times d_v}$ (default $\mathbf{0}$). Chunk size C (assume $C \mid L$). ▷ Projector-only case $\lambda_t = 0$.

Reshape inputs into $M = L/C$ chunks: $\mathbf{U}_{\text{raw}}^{(k)}, \mathbf{Q}^{(k)} \in \mathbb{R}^{d_x \times C}$ and $\mathbf{V}^{(k)} \in \mathbb{R}^{C \times d_v}$, with $\boldsymbol{\eta}^{(k)} \in \mathbb{R}^C$.
▷ Here $\mathbf{x}_t$ is the shifted write-feature stream.
▷ **Phase 1: Intra-chunk structural pre-computation (Parallel)**
for $k = 1$ **to** M **in parallel do**
  $\mathbf{U}^{(k)} \leftarrow \mathbf{U}_{\text{raw}}^{(k)} \cdot \text{diag}(\sqrt{\boldsymbol{\eta}^{(k)}})$ ▷ Scale key-columns
  $\widetilde{\mathbf{V}}^{(k)} \leftarrow \text{diag}(\sqrt{\boldsymbol{\eta}^{(k)}})\, \mathbf{V}^{(k)}$ ▷ Scale value-rows
  $\mathbf{G}^{(k)} \leftarrow {\mathbf{U}^{(k)}}^\top \mathbf{U}^{(k)}$ ▷ Gram Matrix
  $\mathbf{L}^{(k)} \leftarrow \text{tril}(\mathbf{G}^{(k)}, -1) + \mathbf{I}$ ▷ Implicit $\mathbf{T}^{-1}$
end for
▷ **Phase 2: Inter-chunk Recurrence (Sequential)**
$\mathbf{S}_{\text{in}}^{(1)} \leftarrow \mathbf{S}_{\text{init}}$
for $k = 1$ **to** M **do**
  $\mathbf{P}^{(k)} \leftarrow {\mathbf{U}^{(k)}}^\top \mathbf{S}_{\text{in}}^{(k)}$
  $\mathbf{R}^{(k)} \leftarrow \widetilde{\mathbf{V}}^{(k)} - \mathbf{P}^{(k)}$
  $\mathbf{B}^{(k)} \leftarrow \text{solve\_triangular}(\mathbf{L}^{(k)}, \mathbf{R}^{(k)})$ ▷ single residual solve
  $\mathbf{S}_{\text{in}}^{(k+1)} \leftarrow \mathbf{S}_{\text{in}}^{(k)} + \mathbf{U}^{(k)} \mathbf{B}^{(k)}$
end for
$\mathbf{S}_{\text{final}} \leftarrow \mathbf{S}_{\text{in}}^{(M+1)}$ ▷ Final state after processing length L
▷ **Phase 3: Materialize Outputs (Parallel)**
for $k = 1$ **to** M **in parallel do**
  $\mathbf{H}^{(k)} \leftarrow {\mathbf{Q}^{(k)}}^\top \mathbf{S}_{\text{in}}^{(k)}$ ▷ $C \times d_v$: Query interaction with incoming state
  $\boldsymbol{\Lambda}^{(k)} \leftarrow {\mathbf{Q}^{(k)}}^\top \mathbf{U}^{(k)}$ ▷ $C \times C$: Query-Key interaction matrix
  $\mathbf{M}^{(k)} \leftarrow \text{tril}(\boldsymbol{\Lambda}^{(k)}, 0)$ ▷ Causal scores including diagonal
  $\mathbf{O}^{(k)} \leftarrow \mathbf{H}^{(k)} + \mathbf{M}^{(k)} \mathbf{B}^{(k)}$
end for
return $(\mathbf{O}, \mathbf{S}_{\text{final}})$

Single residual solve. The two-solve decomposition is algebraically unnecessary because the local value path and the projected incoming-state path share the same unit-lower-triangular matrix. Let

$$\mathbf{P}^{(k)} := \mathbf{U}^{(k)\top}\mathbf{S}_{\text{in}}^{(k)}, \qquad \mathbf{R}^{(k)} := \widetilde{\mathbf{V}}^{(k)} - \mathbf{P}^{(k)}, \qquad \mathbf{B}^{(k)} := \mathbf{L}^{(k)-1}\mathbf{R}^{(k)}.$$

By linearity,

$$\mathbf{B}^{(k)} = \mathbf{L}^{(k)-1}\widetilde{\mathbf{V}}^{(k)} - \mathbf{L}^{(k)-1}\mathbf{P}^{(k)},$$

so $\mathbf{B}^{(k)}$ is exactly the difference between the older "intra" and "history" solves. Hence

$$\mathbf{S}_{\text{out}}^{(k)} = \mathbf{S}_{\text{in}}^{(k)} + \mathbf{U}^{(k)}\mathbf{B}^{(k)}, \qquad \mathbf{O}^{(k)} = \mathbf{H}^{(k)} + \mathbf{M}^{(k)}\mathbf{B}^{(k)},$$

which removes one triangular solve per chunk in the forward pass without changing the recurrence.

E.3 Chunk-Parallel Backward Pass

Algorithm 7 Parallel DeltaNet Training (Backward, $\lambda_t = 0$)

Require: Upstream gradient $\bar{\mathbf{O}} \in \mathbb{R}^{L\times d_v}$ and cached forward tensors from Algorithm 6 (projector-only, $\lambda_t = 0$) for each chunk $k \in \{1,\dots,M\}$: unscaled keys $\mathbf{U}_{\text{raw}}^{(k)} \in \mathbb{R}^{d\times C}$, queries $\mathbf{Q}^{(k)} \in \mathbb{R}^{d\times C}$, values $\mathbf{V}^{(k)} \in \mathbb{R}^{C\times d_v}$, step sizes $\boldsymbol{\eta}^{(k)} \in \mathbb{R}^{C}$, scaled keys $\mathbf{U}^{(k)} = \mathbf{U}_{\text{raw}}^{(k)}\operatorname{diag}(\sqrt{\boldsymbol{\eta}^{(k)}})$, scaled values $\widetilde{\mathbf{V}}^{(k)} = \operatorname{diag}(\sqrt{\boldsymbol{\eta}^{(k)}})\mathbf{V}^{(k)}$, $\mathbf{L}^{(k)} \in \mathbb{R}^{C\times C}$, residual solves $\mathbf{B}^{(k)} = \text{solve\_triangular}\big(\mathbf{L}^{(k)}, \widetilde{\mathbf{V}}^{(k)} - \mathbf{U}^{(k)\top}\mathbf{S}_{\text{in}}^{(k)}\big)$, and boundary states $\mathbf{S}_{\text{in}}^{(k)} \in \mathbb{R}^{d\times d_v}$.

Ensure: Gradients $\bar{\mathbf{Q}} \in \mathbb{R}^{L\times d}$, $\bar{\mathbf{U}}_{\text{raw}} \in \mathbb{R}^{L\times d}$, $\bar{\mathbf{V}} \in \mathbb{R}^{L\times d_v}$, $\bar{\boldsymbol{\eta}} \in \mathbb{R}^{L}$, and $\bar{\mathbf{S}}_{\text{init}} \in \mathbb{R}^{d\times d_v}$.

Reshape $\bar{\mathbf{O}}$ into chunks $\bar{\mathbf{O}}^{(k)} \in \mathbb{R}^{C\times d_v}$, $M = L/C$.
Initialize $\bar{\mathbf{Q}}^{(k)}, \bar{\mathbf{U}}^{(k)}, \bar{\widetilde{\mathbf{V}}}^{(k)}, \bar{\mathbf{L}}^{(k)}, \bar{\mathbf{S}}_{\text{in}}^{(k)}, \bar{\mathbf{B}}^{(k)}, \bar{\mathbf{P}}^{(k)}, \bar{\mathbf{R}}^{(k)} \leftarrow \mathbf{0}$ for all k.
Set $\bar{\mathbf{S}}_{\text{in}}^{(M+1)}$ to the upstream gradient on the final chunk-exit state (default $\mathbf{0}$ if unused by the loss).
▷ **Adjoint identity (triangular solve).** Let L be unit-lower triangular with fixed diagonal. If $X = L^{-1}B$,
▷ then $\bar{B} = L^{-T}\bar{X}$ and $\bar{L} \mathrel{-}= \operatorname{tril}(\bar{B}X^{\top}, -1)$.
▷ **Adjoint identity (Gram).** If $G = U^{\top}U$, then $\bar{U} \leftarrow \bar{U} + U(\bar{G} + \bar{G}^{\top})$.
▷ **Phase 3$^{\top}$: Backprop through output materialization (Parallel)**
for $k = 1$ **to** M **in parallel do**
▷ Projector-only identity: $\mathbf{O}^{(k)} = \mathbf{H}^{(k)} + \mathbf{M}^{(k)}\mathbf{B}^{(k)}$ with $\mathbf{H}^{(k)} = (\mathbf{Q}^{(k)})^{\top}\mathbf{S}_{\text{in}}^{(k)}$ and $\mathbf{M}^{(k)} = \operatorname{tril}((\mathbf{Q}^{(k)})^{\top}\mathbf{U}^{(k)}, 0)$.
  $\bar{\mathbf{H}}^{(k)} \leftarrow \bar{\mathbf{O}}^{(k)}$
  $\boldsymbol{\Lambda}^{(k)} \leftarrow (\mathbf{Q}^{(k)})^{\top}\mathbf{U}^{(k)}$
  $\mathbf{M}^{(k)} \leftarrow \operatorname{tril}(\boldsymbol{\Lambda}^{(k)}, 0)$
  $\bar{\mathbf{M}}^{(k)} \leftarrow \bar{\mathbf{O}}^{(k)}(\mathbf{B}^{(k)})^{\top}$
  $\bar{\mathbf{M}}^{(k)} \leftarrow \operatorname{tril}(\bar{\mathbf{M}}^{(k)}, 0)$ ▷ $\mathbf{M}^{(k)}$ is causal (includes diagonal)
  $\bar{\mathbf{B}}^{(k)} \leftarrow \bar{\mathbf{B}}^{(k)} + (\mathbf{M}^{(k)})^{\top}\bar{\mathbf{O}}^{(k)}$
▷ Backprop $\mathbf{H}^{(k)} = (\mathbf{Q}^{(k)})^{\top}\mathbf{S}_{\text{in}}^{(k)}$
  $\bar{\mathbf{Q}}^{(k)} \leftarrow \bar{\mathbf{Q}}^{(k)} + \mathbf{S}_{\text{in}}^{(k)}(\bar{\mathbf{H}}^{(k)})^{\top}$
  $\bar{\mathbf{S}}_{\text{in}}^{(k)} \leftarrow \bar{\mathbf{S}}_{\text{in}}^{(k)} + \mathbf{Q}^{(k)}\bar{\mathbf{H}}^{(k)}$
▷ Backprop $\mathbf{M}^{(k)} = \operatorname{tril}(\boldsymbol{\Lambda}^{(k)}, 0)$ and $\boldsymbol{\Lambda}^{(k)} = (\mathbf{Q}^{(k)})^{\top}\mathbf{U}^{(k)}$
  $\bar{\boldsymbol{\Lambda}}^{(k)} \leftarrow \bar{\mathbf{M}}^{(k)}$
  $\bar{\mathbf{Q}}^{(k)} \leftarrow \bar{\mathbf{Q}}^{(k)} + \mathbf{U}^{(k)}(\bar{\boldsymbol{\Lambda}}^{(k)})^{\top}$
  $\bar{\mathbf{U}}^{(k)} \leftarrow \bar{\mathbf{U}}^{(k)} + \mathbf{Q}^{(k)}\bar{\boldsymbol{\Lambda}}^{(k)}$
end for

▷ **Phase $2^\top$: Backprop through inter-chunk recurrence (Sequential reverse)**
for $k = M$ **down to** 1 **do**
  $\bar{\mathbf{S}}^{(k)}_{\text{next}} \leftarrow \bar{\mathbf{S}}^{(k+1)}_{\text{in}}$
  ▷ State update: $\mathbf{S}^{(k+1)}_{\text{in}} = \mathbf{S}^{(k)}_{\text{in}} + \mathbf{U}^{(k)}\mathbf{B}^{(k)}$
  $\bar{\mathbf{S}}^{(k)}_{\text{in}} \leftarrow \bar{\mathbf{S}}^{(k)}_{\text{in}} + \bar{\mathbf{S}}^{(k)}_{\text{next}}$
  $\bar{\mathbf{U}}^{(k)} \leftarrow \bar{\mathbf{U}}^{(k)} + \bar{\mathbf{S}}^{(k)}_{\text{next}}(\mathbf{B}^{(k)})^\top$
  $\bar{\mathbf{B}}^{(k)} \leftarrow \bar{\mathbf{B}}^{(k)} + (\mathbf{U}^{(k)})^\top \bar{\mathbf{S}}^{(k)}_{\text{next}}$
  ▷ Solve: $\mathbf{B}^{(k)} = \text{solve\_triangular}(\mathbf{L}^{(k)}, \mathbf{R}^{(k)})$
  $\bar{\mathbf{R}}^{(k)}_{\text{solve}} \leftarrow \text{solve\_triangular}(\mathbf{L}^{(k)\top}, \bar{\mathbf{B}}^{(k)})$
  $\bar{\mathbf{R}}^{(k)} \leftarrow \bar{\mathbf{R}}^{(k)} + \bar{\mathbf{R}}^{(k)}_{\text{solve}}$
  $\bar{\mathbf{L}}^{(k)} \leftarrow \bar{\mathbf{L}}^{(k)} - \text{tril}\left(\bar{\mathbf{R}}^{(k)}_{\text{solve}}(\mathbf{B}^{(k)})^\top, -1\right)$
  ▷ Residual input: $\mathbf{R}^{(k)} = \widetilde{\mathbf{V}}^{(k)} - \mathbf{P}^{(k)}$
  $\bar{\widetilde{\mathbf{V}}}^{(k)} \leftarrow \bar{\widetilde{\mathbf{V}}}^{(k)} + \bar{\mathbf{R}}^{(k)}$
  $\bar{\mathbf{P}}^{(k)} \leftarrow \bar{\mathbf{P}}^{(k)} - \bar{\mathbf{R}}^{(k)}$
  ▷ Projected incoming state: $\mathbf{P}^{(k)} = (\mathbf{U}^{(k)})^\top \mathbf{S}^{(k)}_{\text{in}}$
  $\bar{\mathbf{U}}^{(k)} \leftarrow \bar{\mathbf{U}}^{(k)} + \mathbf{S}^{(k)}_{\text{in}}(\bar{\mathbf{P}}^{(k)})^\top$
  $\bar{\mathbf{S}}^{(k)}_{\text{in}} \leftarrow \bar{\mathbf{S}}^{(k)}_{\text{in}} + \mathbf{U}^{(k)}\bar{\mathbf{P}}^{(k)}$
end for
$\bar{\mathbf{S}}_{\text{init}} \leftarrow \bar{\mathbf{S}}^{(1)}_{\text{in}}$
▷ **Phase $1^\top$: Backprop through local structure and input scaling (Parallel)**
for $k = 1$ **to** M **in parallel do**
  $\bar{\mathbf{L}}^{(k)} \leftarrow \text{tril}(\bar{\mathbf{L}}^{(k)}, -1)$ ▷ diag of $\mathbf{L}^{(k)}$ is constant ($= \mathbf{I}$)
  $\bar{\mathbf{G}}^{(k)} \leftarrow \bar{\mathbf{L}}^{(k)}$ ▷ since $\mathbf{L}^{(k)} = \text{tril}(\mathbf{G}^{(k)}, -1) + \mathbf{I}$
  $\bar{\mathbf{U}}^{(k)} \leftarrow \bar{\mathbf{U}}^{(k)} + \mathbf{U}^{(k)}\left(\bar{\mathbf{G}}^{(k)} + (\bar{\mathbf{G}}^{(k)})^\top\right)$ ▷ backprop $\mathbf{G}^{(k)} = (\mathbf{U}^{(k)})^\top \mathbf{U}^{(k)}$
  $\boldsymbol{\sigma}^{(k)} \leftarrow \sqrt{\boldsymbol{\eta}^{(k)}}$
  ▷ Unscale: $\mathbf{U}^{(k)} = \mathbf{U}^{(k)}_{\text{raw}} \,\text{diag}(\boldsymbol{\sigma}^{(k)})$ and $\widetilde{\mathbf{V}}^{(k)} = \text{diag}(\boldsymbol{\sigma}^{(k)})\mathbf{V}^{(k)}$
  $\bar{\mathbf{U}}^{(k)}_{\text{raw}} \leftarrow \bar{\mathbf{U}}^{(k)} \,\text{diag}(\boldsymbol{\sigma}^{(k)})$
  $\bar{\mathbf{V}}^{(k)} \leftarrow \text{diag}(\boldsymbol{\sigma}^{(k)})\,\bar{\widetilde{\mathbf{V}}}^{(k)}$
  ▷ Column/row dot definitions: $(\text{col\_dot}(A, B))_i = \langle A_{:,i}, B_{:,i}\rangle$, $(\text{row\_dot}(A, B))_i = \langle A_{i,:}, B_{i,:}\rangle$.
  $\bar{\boldsymbol{\sigma}}^{(k)} \leftarrow \text{col\_dot}(\mathbf{U}^{(k)}_{\text{raw}}, \bar{\mathbf{U}}^{(k)}) + \text{row\_dot}(\mathbf{V}^{(k)}, \bar{\widetilde{\mathbf{V}}}^{(k)})$
  $\mathbf{m}^{(k)} \leftarrow \mathbb{I}[\boldsymbol{\sigma}^{(k)} > 0]$ ▷ element-wise mask
  $\bar{\boldsymbol{\eta}}^{(k)} \leftarrow \mathbf{m}^{(k)} \odot \left(\bar{\boldsymbol{\sigma}}^{(k)} \,/\, (2\boldsymbol{\sigma}^{(k)})\right)$ ▷ Safe when some $\eta_t = 0$ (e.g., boundary $\mathbf{x}_t = \mathbf{0}$): set $\bar{\eta}_t = 0$ instead of forming $0/0$.
end for
return $\bar{\mathbf{Q}}, \bar{\mathbf{U}}_{\text{raw}}, \bar{\mathbf{V}}, \bar{\boldsymbol{\eta}}, \bar{\mathbf{S}}_{\text{init}}$ (reshape chunk grads back to length L).

F Falcon-2 Parallel Implementation

This appendix details the parallel implementation of **Falcon-2** in the multi-head form used in our experiments and codebase. We write the derivation for a single head, the full multi-head attention (MHA) layer applies the same update independently to each head. Unless explicitly stated otherwise, the symbols d and d_v in this appendix therefore denote the per-head key/query feature dimension and value dimension, respectively. Within one head, Falcon-2 introduces column-wise adaptive learning rates $\boldsymbol{\eta}_t$. This implies that every value channel $j \in \{1, \ldots, d_v\}$ follows a unique trajectory in the head-local state space, seemingly preventing the use of shared block-transition matrices.

However, we observe that while the update learning rate varies per column, the geometry of

the updates is determined solely by the keys $\mathbf{k}_t$, which are shared across all value channels within the head. By exploiting this structure, we can vectorize the chunk-wise recurrence while retaining per-dimension adaptivity.

F.1 Per-Channel Dynamics and Shared Geometry

For one head, let $\mathbf{S}_t \in \mathbb{R}^{d\times d_v}$ be the head-local state. The Falcon-2 update for the j-th column $\mathbf{s}_{t,j}$ is:

$$\begin{aligned}\mathbf{s}_{t,j} &= \mathbf{s}_{t-1,j} + \eta_{t,j}\,\mathbf{k}_{t-1}\Big(v_{t,j} - \langle \mathbf{k}_{t-1}, \mathbf{s}_{t-1,j}\rangle\Big)\\ &= \Big(\mathbf{I} - \eta_{t,j}\,\mathbf{k}_{t-1}\mathbf{k}_{t-1}^\top\Big)\mathbf{s}_{t-1,j} + \eta_{t,j}\,v_{t,j}\,\mathbf{k}_{t-1},\end{aligned} \tag{F.1}$$

where we assume $\eta_{t,j} \geq 0$ so that $\sqrt{\eta_{t,j}}$ is well-defined. This makes explicit that the rank-one update matrix $\mathbf{I} - \eta_{t,j}\mathbf{k}_{t-1}\mathbf{k}_{t-1}^\top$ depends on the per-channel step size $\eta_{t,j}$, which determines the chunk-local WY system. To parallelize this over a chunk of size C, we utilize the dual (Gram) formulation.

Projector-only form. For clarity, we present the projector-only case (no explicit per-column decay factor). When ridge/weight decay is enabled in the main text, the unclamped j-th column update includes a scalar decay $\gamma_{j,t} := 1 - \lambda_t\eta_{j,t}$:

$$\mathbf{s}_{t,j} = (\gamma_{t,j}\mathbf{I} - \eta_{t,j}\mathbf{k}_{t-1}\mathbf{k}_{t-1}^\top)\mathbf{s}_{t-1,j} + \eta_{t,j}v_{t,j}\mathbf{k}_{t-1}, \qquad \gamma_{t,j} := 1 - \lambda_t\eta_{t,j}.$$

The implementation uses the clamped positive-decay version described below whenever this unclamped carry is not guaranteed positive. Since each value channel evolves independently, this decay can be removed by a channel-wise cumulative rescaling (equivalently, right-multiplying $\mathbf{S}_t$ by a diagonal matrix of inverse cumulative decays), reducing back to the projector-only form; cf. Appendix G for the shared-decay special case.

Expanded edit decomposition. Expanding the compact main-text update gives

$$\mathbf{S}_t = \mathbf{S}_{t-1}\Big(\mathbf{I}_{d_v} - \lambda_t\,\mathrm{Diag}(\boldsymbol{\eta}_t)\Big) - \mathbf{x}_t\Big(\mathbf{x}_t^\top\mathbf{S}_{t-1}\,\mathrm{Diag}(\boldsymbol{\eta}_t)\Big) + \mathbf{x}_t(\boldsymbol{\eta}_t \odot \mathbf{y}_t)^\top.$$

This isolates the per-column right-multiplicative shrinkage from the feature-direction edit, but it is algebraically identical to the compact update in the main text.

Shared Gram Matrix. Let $\mathbf{K} \in \mathbb{R}^{d\times C}$ be the matrix of (shifted) write keys in the current chunk. The core correlation structure is given by the Gram matrix $\mathbf{G} = \mathbf{K}^\top\mathbf{K} \in \mathbb{R}^{C\times C}$. Crucially, $\mathbf{G}$ is independent of the value dimension d_v and needs to be computed only once per chunk. For channel-wise step sizes, the actual triangular system differs per channel via a simple element-wise modulation of this shared base Gram.

Batched Triangular Solve. The variations in per-channel step sizes $\eta_{t,j}$ modulate this Gram matrix. For each channel j, the implicit system matrix $\mathbf{L}_j$ for the WY representation is:

$$\mathbf{L}_j = \mathrm{tril}\left(\mathbf{G} \odot (\sqrt{\boldsymbol{\eta}}_j\sqrt{\boldsymbol{\eta}}_j^\top), -1\right) + \mathbf{I}_C, \tag{F.2}$$

where $\sqrt{\boldsymbol{\eta}}_j \in \mathbb{R}^C$ is the vector of step-size square-roots for channel j over the chunk, and $\odot$ denotes element-wise multiplication (broadcasting). Equivalently, letting $\mathbf{U}_j := \mathbf{K}\,\mathrm{diag}(\sqrt{\boldsymbol{\eta}}_j)$, we have

$$\mathbf{G}_j := \mathbf{U}_j^\top\mathbf{U}_j = \mathbf{G} \odot (\sqrt{\boldsymbol{\eta}}_j\sqrt{\boldsymbol{\eta}}_j^\top), \qquad \mathbf{L}_j = \mathrm{tril}(\mathbf{G}_j, -1) + \mathbf{I}_C.$$

Because the chunk size C is typically small (e.g., $C = 64$ or 128), we can efficiently perform a batched triangular solve over the batch dimension d_v.

F.2 Chunk-Parallel Algorithm

Algorithm 1 gives the single-head chunk-wise forward pass in a form directly parallel to the scalar-step-size WY kernel in Appendix E, but batched over value channels and written with the positive-decay renormalization used in the implementation. When $\lambda_t = 0$, the decay factors are identically one, and the algorithm reduces to the projector-only WY/Gram kernel. The only difference from the shared-step-size case is that the unit-lower-triangular system $\mathbf{L}_j$ is channel-dependent (via $\boldsymbol{\eta}_{:,j}$), so the single residual solve is carried out in batch over $j \in \{1, \dots, d_v\}$.

Chunk read/write formulas. Let $\mathbf{S}_{\text{in}} \in \mathbb{R}^{d \times d_v}$ be the incoming state for one head in a chunk, and let $\mathbf{K}, \mathbf{Q} \in \mathbb{R}^{d \times C}$ and $\mathbf{V} \in \mathbb{R}^{C \times d_v}$ be the chunk keys, queries, and values (with causal read-after-write ordering inside the chunk). Define layout convention: inside a chunk, we use the feature-major (column-major) layout $\mathbf{K}, \mathbf{Q} \in \mathbb{R}^{d \times C}$ with tokens along columns. Concretely, if $\mathbf{K}_{(m)}, \mathbf{Q}_{(m)} \in \mathbb{R}^{C \times d}$ are the standard time-major slices for chunk m, then we set $\mathbf{K} := \mathbf{K}_{(m)}^\top$ and $\mathbf{Q} := \mathbf{Q}_{(m)}^\top$. We keep $\mathbf{V} \in \mathbb{R}^{C \times d_v}$ time-major, and similarly $\boldsymbol{\eta} \in \mathbb{R}^{C \times d_v}$ for per-channel step sizes.

$$\mathbf{G} := \mathbf{K}^\top \mathbf{K} \in \mathbb{R}^{C \times C}, \qquad \mathbf{M} := \operatorname{tril}(\mathbf{Q}^\top \mathbf{K}, 0) \in \mathbb{R}^{C \times C}, \qquad \mathbf{H} := \mathbf{Q}^\top \mathbf{S}_{\text{in}} \in \mathbb{R}^{C \times d_v},$$

and the projected incoming state $\mathbf{P} := \mathbf{K}^\top \mathbf{S}_{\text{in}} \in \mathbb{R}^{C \times d_v}$. With per-channel step sizes $\boldsymbol{\eta} \in \mathbb{R}^{C \times d_v}$ and $\boldsymbol{\Sigma} := \sqrt{\boldsymbol{\eta}}$, define for each channel j the unit-lower-triangular

$$\mathbf{L}_j := \mathbf{I}_C + \operatorname{tril}\left(\mathbf{G} \odot (\boldsymbol{\Sigma}_{:,j} \boldsymbol{\Sigma}_{:,j}^\top), -1\right).$$

Throughout we include the diagonal (read-after-write) via $\operatorname{tril}(\cdot, 0)$.

Unscaled scores. Note that $\mathbf{M}$ is intentionally formed with the unscaled keys $\mathbf{K}$. All $\sqrt{\eta}$ factors are absorbed into the coefficient matrix $\mathbf{B}$ below; equivalently, for each channel j, $\mathbf{M}(\boldsymbol{\Sigma}_{:,j} \odot \cdot) = \operatorname{tril}\left(\mathbf{Q}^\top \mathbf{K} \operatorname{diag}(\boldsymbol{\Sigma}_{:,j}), 0\right)(\cdot)$.

One-TriSolve reduction. A two-path implementation would solve

$$\mathbf{a}_j := \mathbf{L}_j^{-1}(\boldsymbol{\Sigma}_{:,j} \odot \mathbf{V}_{:,j}), \qquad \mathbf{c}_j := \mathbf{L}_j^{-1}(\boldsymbol{\Sigma}_{:,j} \odot \mathbf{P}_{:,j}),$$

and then form $\mathbf{B}_{:,j} = \boldsymbol{\Sigma}_{:,j} \odot (\mathbf{a}_j - \mathbf{c}_j)$. Because both paths share the same $\mathbf{L}_j$, linearity gives the exact merged solve

$$\widetilde{\mathbf{b}}_j := \mathbf{L}_j^{-1}\Big(\boldsymbol{\Sigma}_{:,j} \odot (\mathbf{V}_{:,j} - \mathbf{P}_{:,j})\Big), \qquad \mathbf{B}_{:,j} := \boldsymbol{\Sigma}_{:,j} \odot \widetilde{\mathbf{b}}_j.$$

This is the form used throughout the paper. It is algebraically identical to the older two-solve decomposition, but removes one batched TriSolve per chunk in the forward pass. The chunk outputs and chunk-exit state are then

$$\mathbf{O} = \mathbf{H} + \mathbf{M}\mathbf{B} \in \mathbb{R}^{C \times d_v}, \qquad \mathbf{S}_{\text{out}} = \mathbf{S}_{\text{in}} + \mathbf{K}\mathbf{B} \in \mathbb{R}^{d \times d_v}.$$

Pseudocode. Algorithm 1 in the main text gives the single consolidated one-TriSolve forward pass; this appendix focuses on the derivation and complexity.

Efficiency Analysis. Per chunk, forming the shared Gram and score matrices costs $\mathcal{O}(dC^2)$ for $\mathbf{K}^\top\mathbf{K}$ and $\mathbf{Q}^\top\mathbf{K}$. Constructing the channel-specific systems $\{\mathbf{L}_j\}$ and performing the single batched residual solve cost $\mathcal{O}(d_vC^2)$. Materializing the chunk outputs via $\mathbf{MB}$ also costs $\mathcal{O}(d_vC^2)$, and the chunk-exit state update $\mathbf{KB}$ costs $\mathcal{O}(d\,d_v\,C)$. Relative to the older two-solve presentation, the asymptotic complexity is unchanged, but the dominant forward triangular-solve count drops from two to one per chunk. Compared to a naive per-channel implementation that would require explicit $d \times d$ operators per value dimension, this confines the rate-dependence to small $C \times C$ systems. Nevertheless, the $\mathcal{O}(d_vC^2)$ terms can still be substantial for large d_v, motivating the shared-dynamics Falcon variant in Appendix G.

Including ridge/weight decay ($\lambda_t > 0$). The main-text Algorithm 1 already includes the $\lambda_t > 0$ path. Algebraically, the decayed recurrence is reduced to the projector-only WY/Gram kernel by a channel-wise cumulative rescaling. With per-channel step sizes $\eta_{t,j}$ and scalar ridge λ_t, the j-th column evolves as

$$\mathbf{s}_{t,j} = ((1 - \lambda_t\eta_{t,j})\mathbf{I} - \eta_{t,j}\mathbf{k}_{t-1}\mathbf{k}_{t-1}^\top)\mathbf{s}_{t-1,j} + \eta_{t,j}v_{t,j}\mathbf{k}_{t-1}.$$

For the unclamped derivation, define the per-channel decay $\gamma_{t,j} := 1-\lambda_t\eta_{t,j}$ and cumulative products $c_{0,j} := 1$, $c_{t,j} := \prod_{s=1}^{t}\gamma_{s,j}$, and $\mathbf{D}_t := \mathrm{Diag}(c_{t,1}, \ldots, c_{t,d_v})$. In the implemented log-space path, $\gamma_{t,j}$ is replaced by the clamped positive carry defined in the note below. Then the column-wise rescaled state $\widetilde{\mathbf{S}}_t := \mathbf{S}_t\mathbf{D}_t^{-1}$ satisfies a projector-only recurrence:

$$\widetilde{\mathbf{s}}_{t,j} = (\mathbf{I} - \widetilde{\eta}_{t,j}\mathbf{k}_{t-1}\mathbf{k}_{t-1}^\top)\widetilde{\mathbf{s}}_{t-1,j} + \widetilde{\eta}_{t,j}\widetilde{v}_{t,j}\mathbf{k}_{t-1}, \qquad \widetilde{\eta}_{t,j} := \eta_{t,j}/\gamma_{t,j}, \qquad \widetilde{v}_{t,j} := v_{t,j}/c_{t-1,j}.$$

Equivalently, the chunk kernel runs the projector-only WY/Gram equations on the rescaled variables $\widetilde{\boldsymbol{\eta}}$ and $\widetilde{\mathbf{V}}$ (working in the $\widetilde{\mathbf{S}}$ domain), and then recovers the original outputs and state by $\mathbf{O}_t = \widetilde{\mathbf{O}}_t\mathbf{D}_t$ and $\mathbf{S}_t = \widetilde{\mathbf{S}}_t\mathbf{D}_t$ (i.e., element-wise multiplication by $c_{t,j}$ on each channel).

Log-space note. First form $\alpha_{t,j} := \lambda_t\eta_{t,j}$, clamp $\alpha_{t,j} \leftarrow \min(\alpha_{t,j}, 1 - \varepsilon_\gamma)$, and then compute $\log\gamma_{t,j} = \mathrm{log1p}(-\alpha_{t,j})$ in fp32 so that $\gamma_{t,j} > 0$. In chunk-wise kernels, do *not* form global products $c_{t,j}$ (or $\exp(\pm\zeta_{t,j})$) over the full sequence; instead, reset the log-prefix at each chunk boundary and use only chunk-local prefixes (Algorithm 1).

Main-text algorithm. The consolidated positive-decay pseudocode appears in Algorithm 1; no separate projector-only pseudocode is needed because setting $\lambda_t = 0$ makes all local decay prefixes equal to one.

G Falcon with Shared Dynamics

Appendix F derives the full per-channel Falcon-2 parallelization. That formulation is expressive but requires a distinct step-size trajectory for every value channel $j \in \{1, \ldots, d_v\}$.

This creates a computational bottleneck: the implicit inverse matrix $\mathbf{T}$ (or the Cholesky factor $\mathbf{L}$) depends on the learning rates. Consequently, the full Falcon-2 requires solving d_v distinct triangular systems of size $C \times C$ for every chunk. Although d_v denotes the per-head value dimension in this appendix, the aggregate number of value channels over heads can be large, which prevents the use of a single efficient block solve and leads to high memory bandwidth usage.

In this section, we introduce **Falcon-1**, a hardware-efficient variant that enforces a shared adaptive learning rate across all value channels. This constraint reduces the number of distinct $C \times C$ triangular systems that must be constructed from d_v to 1, enabling a single multi-right-hand-side triangular solve per chunk while keeping the per-step projector dynamics shared across value channels. This shared-dynamics reduction is orthogonal to head factorization: in the practical MHA setting used here, one first applies the per-head factorization and then decides whether the step-size dynamics inside each head are full (per value channel) or shared.

Asymptotically, applying the state update still costs $\mathcal{O}(d\, d_v\, C)$ and solving a triangular system with d_v right-hand sides costs $\mathcal{O}(d_v C^2)$. The primary savings come from eliminating the need to construct and factor d_v distinct $C \times C$ systems: Falcon-1 uses one shared system per chunk and a single high-throughput solve with many right-hand sides.

G.1 Scalar versus Vector Step Sizes

In the full Falcon-2, the state update is driven by a vector rate $\boldsymbol{\eta}_t \in \mathbb{R}^{d_v}$. In Falcon-1, we constrain this to a scalar $\eta_t \in \mathbb{R}$, derived from the global energy of the write feature $\mathbf{x}_t$:

$$\eta_t = \frac{\beta_t}{\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon}, \qquad \beta_t \in (0, 2),\ \lambda_t \geq 0,\ \varepsilon \geq 0, \tag{G.1}$$

where β_t is a learnable scalar NLMS gain, λ_t is the ridge coefficient (default $\lambda_t > 0$), and ε is a small stabilizer (cf. Eq. (3.4)). The autoregressive update rule for the state matrix $\mathbf{S}_t \in \mathbb{R}^{d \times d_v}$ simplifies to:

$$\mathbf{S}_t = \underbrace{\left((1 - \eta_t \lambda_t)\mathbf{I} - \eta_t \mathbf{x}_t \mathbf{x}_t^\top\right)}_{\mathbf{A}_t} \mathbf{S}_{t-1} + \eta_t\, \mathbf{x}_t \mathbf{v}_t^\top, \qquad \gamma_t^{\text{raw}} := 1 - \eta_t \lambda_t. \tag{G.2}$$

Crucially, the transition matrix $\mathbf{A}_t$ is now *shared* across all columns of $\mathbf{S}_t$. This implies that while different features store different contents (values), they share the same dynamics (write/forget speeds).

Including weight decay. Falcon-2 uses $\lambda_t > 0$ by default, inducing the unclamped shrinkage factor $\gamma_t^{\text{raw}} := 1 - \eta_t \lambda_t$. For the log-space reduction below, we require a positive carry $\gamma_t > 0$ (equivalently, the realized decay fraction must be < 1). Under our default RMSNorm scaling (so $\|\mathbf{x}_t\|_2^2 \approx d$) and $\lambda_t \in [0, 1]$, Eq. (G.1) implies $\eta_t \lambda_t \lesssim \beta_t/(d+1) < 1$ and therefore $\gamma_t \in (0, 1]$ automatically away from the boundary sentinel. More generally (and in finite precision), we clamp the decay fraction $\alpha_t := \eta_t \lambda_t$ as $\alpha_t \leftarrow \min(\alpha_t, 1 - \varepsilon_\gamma)$, i.e. $\gamma_t \leftarrow \max(1 - \eta_t \lambda_t, \varepsilon_\gamma)$, so that $\gamma_t \geq \varepsilon_\gamma > 0$. For the identity-feature Falcon setting, one may realize the boundary no-op via $\mathbf{k}_0 = \mathbf{0}$ and hence $\mathbf{x}_1 = \mathbf{0}$; in the general kernelized setting the mathematically correct convention is imposed directly as $\mathbf{x}_1 := \mathbf{0}$. We take $\eta_1 = 0$ and $\gamma_1 = 1$ (no write/decay), matching the main-text convention that updates start at $t = 2$. Because γ_t is a scalar, one can reduce the decayed recurrence to the projector-only case by rescaling. Let $c_0 := 1$ and $c_t := \prod_{r=1}^{t} \gamma_r$, and define $\widetilde{\mathbf{S}}_t := \mathbf{S}_t / c_t$ (so under read-after-write, reads satisfy $\mathbf{o}_t = c_t \tilde{\mathbf{o}}_t$). Then the decayed update is equivalent to running the projector-only WY kernels on

$$\widetilde{\eta}_t := \eta_t / \gamma_t, \qquad \widetilde{\mathbf{v}}_t := \mathbf{v}_t / c_{t-1},$$

and rescaling outputs by c_t. Algorithms 8-9 implement this reduction explicitly.

Numerical stability. The algebraic reduction to the projector-only WY kernels rescales targets by the inverse cumulative decay, $\widetilde{\mathbf{v}}_t = \mathbf{v}_t / c_{t-1}$ with $c_t = \prod_{r \leq t} \gamma_r$. While correct in exact arithmetic, in finite

precision c_t may become extremely small when γ_t is noticeably below 1, making $c_{t-1}^{-1} = \exp(-\zeta_{t-1})$ overflow and yielding $\infty \times 0$ patterns (and NaNs) when rescaling back by $\exp(\zeta_t)$.

Stable chunk-local renormalization. To avoid ever forming $\exp(\pm\zeta_t)$ over the full sequence, our implementation performs the same reduction within each WY chunk of length C. For a chunk $[a, b]$ (where $b = a+C-1$), define local log-prefix decays $u_0 := 0$ and $u_i := \sum_{r=0}^{i-1} \log \gamma_{a+r}$ for $i = 1, \ldots, C$, with $\delta_i := \exp(u_i)$. We run the projector-only WY kernel on the rescaled values $\widehat{\mathbf{v}}_{a+i-1} := \mathbf{v}_{a+i-1} \exp(-u_{i-1}) = \mathbf{v}_{a+i-1}/\delta_{i-1}$ and step sizes $\widehat{\eta}_t := \eta_t/\gamma_t$ (compute $\log \gamma_t = \mathrm{log1p}(-\alpha_t)$ in fp32 with $\alpha_t := \min(\eta_t \lambda_t, 1 - \varepsilon_\gamma)$, then $\widehat{\eta}_t = \eta_t \exp(-\log \gamma_t)$). Outputs and the chunk-exit state are rescaled back by the *forward* local decays: $\mathbf{o}_{a+i-1} = \delta_i \widehat{\mathbf{o}}_{a+i-1}$ and $\mathbf{S}_b = \delta_C \widehat{\mathbf{S}}_b$. This is algebraically equivalent to the global reduction but bounds exponentials by $O(C)$ rather than $O(L)$.

G.2 Shared-WY Parallelization

Falcon-1 uses a *single scalar* step size η_t shared across all value channels. Consequently, within each chunk, the WY factors are shared across columns of $\mathbf{S}$: the per-chunk system matrix $\mathbf{L}^{(k)}$ is *identical* for all d_v columns. This is precisely the setting of Appendix E (projector-only, $\lambda_t = 0$), where the WY implementation yields one unit-lower-triangular matrix $\mathbf{L}^{(k)} \in \mathbb{R}^{C \times C}$ per chunk and, after merging the write and history paths into a single residual right-hand side, one standard *multi-RHS* triangular solve with d_v right-hand sides.

Forward/backward reuse. With shared dynamics, the chunk-parallel attention forward pass is exactly Algorithm 6 and the backward pass through the WY machinery is Algorithm 7. The only additional ingredient in Falcon-1 is that the step size η_t is not a free input: it is produced by the NLMS normalization

$$\eta_t = \frac{\beta_t}{\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon}, \qquad \mathbf{x}_t := \phi(\mathbf{k}_{t-1}) \text{ is the shifted write feature,} \tag{G.3}$$

and the default $\lambda_t > 0$ introduces the additional scalar pathway $\gamma_t = 1 - \eta_t \lambda_t$. Backpropagation therefore, includes chain-rule steps through both η_t and the cumulative decay c_t (Algorithm 9).

G.3 Forward and Backward Algorithms

Forward. Compute $(\eta_t, \log \gamma_t)$ (with clamping) and run the single-solve projector-only WY kernel on chunk-locally renormalized inputs: within each chunk compute local prefixes u_i and $\delta_i = \exp(u_i)$, use $\widehat{\eta}_t = \eta_t/\gamma_t$ and $\widehat{\mathbf{v}}_t = \mathbf{v}_t/\delta_{i-1}$ inside that chunk, and rescale outputs/state by δ_i (Algorithm 8).

Algorithm 8 Falcon-1 (Forward)

Require: Shifted write features $\mathbf{x} \in \mathbb{R}^{L\times d}$ (boundary $\mathbf{x}_1 = \mathbf{0}$), queries $\mathbf{Q} \in \mathbb{R}^{L\times d}$, values $\mathbf{V} \in \mathbb{R}^{L\times d_v}$, gains $\beta \in (0,2)^L$, ridge $\lambda \in \mathbb{R}^L_{\geq 0}$, stabilizer $\varepsilon > 0$, clamp $\varepsilon_\gamma > 0$, dummy $\varepsilon_\eta > 0$, chunk size C (assume $C \mid L$), initial state $\mathbf{S}_{\text{init}}$.

Ensure: Outputs $\mathbf{O}$ for the decayed recurrence with $\eta_t = \beta_t/(\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon)$, $\alpha_t := \min(\eta_t\lambda_t, 1-\varepsilon_\gamma)$, and $\gamma_t := 1 - \alpha_t > 0$. If the clamp is active, the shrinkage path corresponds to the effective coefficient α_t/η_t for $\eta_t > 0$.

Compute $\eta_1 \leftarrow 0$, and for $t \geq 2$: $d_t \leftarrow \|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon$, $\eta_t \leftarrow \beta_t/d_t$ (fp32 for norms/ratios).
$\alpha_t^{\text{raw}} \leftarrow \eta_t\lambda_t$, $\alpha_t \leftarrow \min(\alpha_t^{\text{raw}}, 1-\varepsilon_\gamma)$ for $t \geq 2$.
$\log\gamma_1 \leftarrow 0$ and for $t \geq 2$: $\log\gamma_t \leftarrow \operatorname{log1p}(-\alpha_t)$ (fp32), $\gamma_t \leftarrow \exp(\log\gamma_t)$.
$\widehat{\eta}_1 \leftarrow \varepsilon_\eta$ (dummy; no effect since $\mathbf{x}_1 = \mathbf{0}$), and for $t \geq 2$: $\widehat{\eta}_t \leftarrow \eta_t/\gamma_t = \eta_t \exp(-\log\gamma_t)$.
Partition into $M = L/C$ chunks $[a_k, b_k] = [(k-1)C+1, kC]$. Set $\mathbf{S}_{\text{in}} \leftarrow \mathbf{S}_{\text{init}}$.
for $k = 1, \ldots, M$ **do**
  $a \leftarrow a_k$, $b \leftarrow b_k$.
  $u_0 \leftarrow 0$ ▷ Local log-prefix decays inside the chunk
  for $i = 1, \ldots, C$ **do**
    $u_i \leftarrow u_{i-1} + \log\gamma_{a+i-1}$, $\delta_i \leftarrow \exp(u_i)$ ▷ fp32
  end for
  $\widehat{\mathbf{V}}_{a+i-1} \leftarrow \mathbf{V}_{a+i-1}\exp(-u_{i-1})$ for $i = 1, \ldots, C$ ▷ $\widehat{\mathbf{v}} = \mathbf{v}/\delta_{i-1}$
  Run the single-solve projector-only WY kernel on the length-C sequence $(\mathbf{x}_{a:b}, \mathbf{Q}_{a:b}, \widehat{\mathbf{V}}_{a:b}, \widehat{\eta}_{a:b})$ with init $\mathbf{S}_{\text{in}}$ (Algorithm 6 with $L = C$), producing $(\widehat{\mathbf{O}}_{a:b}, \widehat{\mathbf{S}}_{\text{out}})$.
  $\mathbf{O}_{a+i-1} \leftarrow \delta_i\,\widehat{\mathbf{O}}_{a+i-1}$ for $i = 1, \ldots, C$.
  $\mathbf{S}_{\text{in}} \leftarrow \delta_C\,\widehat{\mathbf{S}}_{\text{out}}$ ▷ Propagate original chunk-exit state
end for
return $\mathbf{O}$.

Backward. Algorithm 7 provides gradients for the one-solve projector-only kernel while treating $(\mathbf{x}, \boldsymbol{\eta})$ as independent. Falcon additionally backpropagates through the NLMS map $\eta_t = \beta_t/(\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon)$ and through the chunk-local log-decay renormalization (the local prefixes u_i and δ_i) used to handle γ_t (Algorithm 9).

Algorithm 9 Falcon-1 (Backward)

Require: Upstream gradient $\bar{\mathbf{O}}$, chunk-local caches from Algorithm 8 for each chunk k: local prefixes $\{u_i, \delta_i\}_{i=0}^C$, $\log\gamma_{a_k:b_k}$, (η, β, λ), clamp mask $m_t = \mathbb{I}[\eta_t\lambda_t < 1 - \varepsilon_\gamma]$, and WY caches from running the projector-only kernel on $(\mathbf{x}_{a_k:b_k}, \mathbf{Q}_{a_k:b_k}, \widehat{\mathbf{V}}_{a_k:b_k}, \widehat{\eta}_{a_k:b_k})$.

Ensure: Gradients $(\bar{\mathbf{Q}}, \bar{\mathbf{x}}, \bar{\mathbf{V}}, \bar{\beta}, \bar{\lambda}, \bar{\mathbf{S}}_{\text{init}})$.

Initialize all gradients to zero. Set boundary $\bar{\mathbf{S}}_{\text{in}}^{(M+1)} \leftarrow \mathbf{0}$.
for $k = M, \ldots, 1$ **do** ▷ reverse over chunks
  $a \leftarrow a_k$, $b \leftarrow b_k$.
  Initialize $\bar{u}_i \leftarrow 0$ for $i = 0, \ldots, C$, $\bar{\log\gamma}_t \leftarrow 0$ and $\bar{\eta}_t \leftarrow 0$ for $t = a, \ldots, b$.
  ▷ Unscale outputs: $\mathbf{O}_t = \delta_i\,\widehat{\mathbf{O}}_t$
  for $t = a, \ldots, b$ **do**
    $i \leftarrow t - a + 1$
    $\bar{\widehat{\mathbf{O}}}_t \leftarrow \delta_i\,\bar{\mathbf{O}}_t$
    $\bar{u}_i \mathrel{+}= \langle \bar{\widehat{\mathbf{O}}}_t, \widehat{\mathbf{O}}_t \rangle$ ▷ $\partial\delta_i/\partial u_i = \delta_i$

end for

▷ Boundary: next chunk sees $\mathbf{S}_{\text{in}}^{(k+1)} = \delta_C \, \widehat{\mathbf{S}}_{\text{out}}$

$\bar{\widehat{\mathbf{S}}}_{\text{out}} \leftarrow \delta_C \, \bar{\mathbf{S}}_{\text{in}}^{(k+1)}$

$\bar{u}_C \mathrel{+}= \langle \bar{\widehat{\mathbf{S}}}_{\text{out}}, \widehat{\mathbf{S}}_{\text{out}} \rangle$

Run the projector-only WY backward on this length-C chunk (Algorithm 7 with $L = C$), using upstream $(\bar{\widehat{\mathbf{O}}}_{a:b}, \bar{\widehat{\mathbf{S}}}_{\text{out}})$, to obtain $(\bar{\mathbf{Q}}_{a:b}, \bar{\mathbf{x}}_{a:b}, \bar{\widehat{\mathbf{V}}}_{a:b}, \bar{\widehat{\eta}}_{a:b}, \bar{\mathbf{S}}_{\text{in}}^{(k)})$.

▷ Unscale values: $\widehat{\mathbf{V}}_t = \mathbf{V}_t \exp(-u_{i-1})$

for $t = a, \ldots, b$ **do**

$i \leftarrow t - a + 1$

$s \leftarrow \exp(-u_{i-1})$

$\bar{\mathbf{V}}_t \mathrel{+}= s \, \bar{\widehat{\mathbf{V}}}_t$

$\bar{u}_{i-1} \mathrel{+}= -\langle s \, \bar{\widehat{\mathbf{V}}}_t, \mathbf{V}_t \rangle$

end for

▷ Unscale step sizes: $\widehat{\eta}_t = \eta_t \exp(-\log \gamma_t)$

for $t = \max(a, 2), \ldots, b$ **do**

$g^{-1} \leftarrow \exp(-\log \gamma_t)$

$\bar{\eta}_t \mathrel{+}= g^{-1} \, \bar{\widehat{\eta}}_t$

$\overline{\log \gamma}_t \mathrel{+}= -(\eta_t \, g^{-1}) \, \bar{\widehat{\eta}}_t$

end for

▷ Backprop local prefix sums: $u_i = u_{i-1} + \log \gamma_{a+i-1}$

for $i = C$ **down to** 1 **do**

$t \leftarrow a + i - 1$

if $t \geq 2$ **then**

$\overline{\log \gamma}_t \mathrel{+}= \bar{u}_i$

end if

$\bar{u}_{i-1} \mathrel{+}= \bar{u}_i$

end for

▷ Backprop $\log \gamma_t = \mathrm{log1p}(-\alpha_t)$ with $\alpha_t = \min(\eta_t \lambda_t, 1 - \varepsilon_\gamma)$

for $t = \max(a, 2), \ldots, b$ **do**

$\bar{\alpha}_t \leftarrow -\exp(-\log \gamma_t) \, \overline{\log \gamma}_t$

$\bar{\alpha}_t \leftarrow m_t \, \bar{\alpha}_t$ ▷ zero gradient when clamped

$\bar{\eta}_t \mathrel{+}= \lambda_t \, \bar{\alpha}_t$

$\bar{\lambda}_t \mathrel{+}= \eta_t \, \bar{\alpha}_t$

end for

▷ Backprop $\eta_t = \beta_t / (\|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon)$

for $t = \max(a, 2), \ldots, b$ **do**

$d_t \leftarrow \|\mathbf{x}_t\|_2^2 + \lambda_t + \varepsilon$

$\bar{\beta}_t \mathrel{+}= \bar{\eta}_t / d_t$

$\bar{d}_t \leftarrow -\beta_t \, \bar{\eta}_t / d_t^2$

$\bar{\mathbf{x}}_t \mathrel{+}= 2 \, \bar{d}_t \, \mathbf{x}_t$

$\bar{\lambda}_t \mathrel{+}= \bar{d}_t$

end for

▷ Boundary constants $\widehat{\eta}_1 := \varepsilon_\eta$ and $\log \gamma_1 := 0$ carry no gradient.

end for

return $(\bar{\mathbf{Q}}, \bar{\mathbf{x}}, \bar{\mathbf{V}}, \bar{\beta}, \bar{\lambda}, \bar{\mathbf{S}}_{\text{init}})$ with $\bar{\mathbf{S}}_{\text{init}} = \bar{\mathbf{S}}_{\text{in}}^{(1)}$.

G.4 Complexity Analysis

Table 4 summarizes the per-chunk costs. The residual formulation removes one forward TriSolve per chunk in both Falcon-2 and Falcon-1 kernels, but Falcon-1 retains the larger practical gain because it also shares the system build across all value channels. Sharing the dynamics therefore eliminates the rate-dependent system construction overhead that scales as $\mathcal{O}(d_v C^2)$ in the full Falcon-2 (building/factorizing d_v distinct $C \times C$ triangular systems). The remaining triangular solve still scales as $\mathcal{O}(d_v C^2)$ because the state has d_v columns, but it is a single highly optimized multi-RHS solve, which is substantially more GPU-friendly in practice.

Table 4 Complexity per Chunk (chunk size C, feature dim d, value dim d_v).

Component	Falcon-2 (Full)	Falcon-1
Gram Matrix	$\mathcal{O}(dC^2)$	$\mathcal{O}(dC^2)$
Rate-dependent system build ($\mathbf{L}$)	$\mathcal{O}(\mathbf{d_v C^2})$	$\mathcal{O}(\mathbf{C^2})$
Forward residual TriSolve	$\mathcal{O}(d_v C^2)$	$\mathcal{O}(d_v C^2)$
State update / output projection	$\mathcal{O}(dd_v C)$	$\mathcal{O}(dd_v C)$

Falcon-1 removes the need to construct and factor d_v distinct $C \times C$ systems by enforcing shared dynamics. After the residual merge, the remaining forward solve is a single high-throughput multi-RHS triangular solve, which is substantially more GPU-friendly in practice.

H Falcon-3 ParallelFlow Implementation

Falcon-3 uses a sliding regression window of size B, which turns the rank-one DeltaNet-style update into a rank-B affine recurrence. To train this recurrence without materializing dense state-transition matrices or scanning token by token, we use the ParallelFlow framework (Cirone and Salvi, 2025).

This appendix states the chunk map used in the scan, summarizes the low-rank controlled-differential-equation view, and maps the Falcon-3 update to the `tensorInv` solve used in Algorithm 3.

Mask conventions. Within `tensorInv`, we use a block-strict-causal mask over (time, rank) pairs, so same-time rank components do not interact, and every residual is evaluated at the pre-update state $\mathbf{S}_{t-1}$. Output materialization then uses an inclusive block-causal mask to implement read-after-write inside each chunk.

H.1 Affine Chunk Map

To make the scan claim explicit, define for chunk k

$$\mathbf{M}^{(k)} := \delta_C^{(k)}(\mathbf{I}_{d_x} + \mathbf{A}^{(k)}\mathbf{W}^{(k)}), \qquad \mathbf{b}^{(k)} := \delta_C^{(k)}\mathbf{A}^{(k)}\mathbf{U}^{(k)}.$$

Then the boundary update is the affine map

$$\mathbf{S}_{\text{in}}^{(k+1)} = \mathbf{M}^{(k)}\mathbf{S}_{\text{in}}^{(k)} + \mathbf{b}^{(k)}.$$

These chunk maps compose associatively,

$$(\mathbf{M}_2, \mathbf{b}_2) \circ (\mathbf{M}_1, \mathbf{b}_1) = (\mathbf{M}_2\mathbf{M}_1,\ \mathbf{M}_2\mathbf{b}_1 + \mathbf{b}_2),$$

so Phase 2 of Algorithm 3 may be implemented either as the simple sequential loop shown in the main text or as an associative scan over chunk maps.

H.2 Matrix-Valued CDEs and Low-Rank Drivers

Standard linear recurrences can be viewed as discretizations of a continuous-time process. Let $\mathbf{S}_t \in \mathbb{R}^{d\times d_v}$ be the hidden state. We adopt the left-multiplicative convention and model its evolution as a matrix-valued CDE driven by $\boldsymbol{\omega}$ and $\boldsymbol{\xi}$:

$$d\mathbf{S}_t = d\boldsymbol{\omega}_t \mathbf{S}_t + d\boldsymbol{\xi}_t, \tag{H.1}$$

where $\boldsymbol{\omega}_t \in \mathbb{R}^{d\times d}$ and $\boldsymbol{\xi}_t \in \mathbb{R}^{d\times d_v}$ are matrix-valued paths. The solution over an interval $[s,t]$ factors through a linear propagator (flow) $\mathbf{P}_{t\leftarrow s}$:

$$\mathbf{S}_t = \mathbf{P}_{t\leftarrow s}\mathbf{S}_s + \int_s^t \mathbf{P}_{t\leftarrow r}\, d\boldsymbol{\xi}_r, \quad \text{where } \mathbf{P}_{t\leftarrow s} = \mathbf{I} + \int_s^t d\boldsymbol{\omega}_r \mathbf{P}_{r\leftarrow s}. \tag{H.2}$$

This decomposition separates the temporal dynamics ($\mathbf{P}$) from the computation. ParallelFlow parallelizes this by partitioning the sequence into chunks $[t_{k-1}, t_k]$. The system computes local propagators and accumulated input injections for each chunk in parallel, then links them via a global associative scan.

Transpose-equivalent form. ParallelFlow is often written after transposing the state. Defining $\widehat{\mathbf{S}}_t := \mathbf{S}_t^\top \in \mathbb{R}^{d_v\times d}$, the same low-rank update becomes the shared-right-factor recurrence

$$\widehat{\mathbf{S}}_{t+1} = \widehat{\mathbf{S}}_t + \widehat{\mathbf{S}}_t\, \mathbf{B}_t \mathbf{A}_t^\top + \widetilde{\mathbf{B}}_t \mathbf{A}_t^\top.$$

Transposing back yields the shared-left-factor form

$$\mathbf{S}_{t+1} = \mathbf{S}_t + \mathbf{A}_t \mathbf{B}_t^\top \mathbf{S}_t + \mathbf{A}_t \widetilde{\mathbf{B}}_t^\top.$$

We use the shared-left-factor form throughout this appendix because the sliding regression update naturally shares the window matrix of write features.

Low-Rank Drivers. The core difficulty lies in computing the propagator $\mathbf{P}_{s\to t}$, which is typically an expensive matrix ODE. However, efficient computation is possible if the drivers possess a low-rank structure. We assume rank-R drivers with a shared left factor:

$$d\boldsymbol{\omega}_t = \vec{\mathbf{A}}_t \vec{\mathbf{B}}_t^\top dt, \qquad d\boldsymbol{\xi}_t = \vec{\mathbf{A}}_t \widetilde{\vec{\mathbf{B}}}_t^\top dt, \tag{H.3}$$

with $\vec{\mathbf{A}}_t, \vec{\mathbf{B}}_t \in \mathbb{R}^{d\times R}$ and $\widetilde{\vec{B}}_t \in \mathbb{R}^{d_v\times R}$. This structure allows for an efficiently computable, compact representation of the flow. For discrete tokens, under a forward-Euler discretization (with the step size absorbed into $\vec{B}, \widetilde{\vec{B}}$, equivalently $\Delta t = 1$), the update becomes:

$$\mathbf{S}_{t_{k+1}} = \mathbf{S}_{t_k} + \vec{\mathbf{A}}_{t_k} \vec{\mathbf{B}}_{t_k}^\top \mathbf{S}_{t_k} + \vec{\mathbf{A}}_{t_k} \widetilde{\vec{\mathbf{B}}}_{t_k}^\top. \tag{H.4}$$

The `tensorInv` Algorithm. Solving this low-rank system over a chunk of length L_c can be reduced to a Triangular Tensor Inversion. Let $\mathcal{C} \in \mathbb{R}^{(L_c \times R) \times (L_c \times R)}$ be a tensor representing causal interactions between rank-components. For indices $s, t \in \{1, \ldots, L_c\}$ and ranks $i, j \in \{1, \ldots, R\}$:

$$[\mathcal{C}]_{t,i}^{s,j} = \delta_{s,t}\delta_{i,j} - \mathbb{I}(s < t) \left[\vec{\mathbf{B}}_t^\top \vec{\mathbf{A}}_s\right]_{i,j}. \tag{H.5}$$

Cirone and Salvi (2025) shows that the propagator can be computed by applying the structured inverse tensor $\mathcal{D} = \mathcal{C}^{-1}$ under the tensor-contraction product, thereby avoiding explicit dense $d_x \times d_x$ chunk propagators. In our setting, the computation is governed by an $(L_c R) \times (L_c R)$ structured causal solve together with matrix multiplications involving d_x and d_v; the compressed asymptotic $\mathcal{O}(L_c^2 R + d)$ is therefore misleading here because it hides the dominant dimension-dependent terms.

`tensorInv` outputs (the objects used in Algorithm 3). In practice, we do not materialize the full inverse tensor $\mathcal{D}$. Instead, ParallelFlow applies $\mathcal{C}^{-1}$ to two right-hand sides corresponding to the low-rank drivers. For a chunk of length L_c and rank R, stack (flatten time×rank) the discrete drivers as

$$\vec{\mathbf{A}} \in \mathbb{R}^{d_x \times (L_c R)}, \qquad \vec{\mathbf{B}} \in \mathbb{R}^{d_x \times (L_c R)}, \qquad \widetilde{\vec{B}} \in \mathbb{R}^{d_v \times (L_c R)}.$$

Let $\vec{M} \in \{0, 1\}^{(L_c R) \times (L_c R)}$ denote the strictly-causal block mask that is 1 for interactions from earlier times $s < t$ and 0 otherwise; equivalently, in (t, i) indexing it is exactly the indicator $\mathbb{I}(s < t)$ used in the definition of $\mathcal{C}$ above. In particular, after flattening time×rank, same-time rank components do not interact inside the triangular solve. Then the chunk-local triangular system can be written compactly as

$$\mathcal{C} = \mathbf{Id} - \vec{M} \odot (\vec{B}^\top \vec{A}),$$

and the two solves returned by `tensorInv` are

$$\vec{W} := \mathcal{C}^{-1} \vec{B}^\top \in \mathbb{R}^{(L_c R) \times d_x}, \qquad \vec{U} := \mathcal{C}^{-1} \widetilde{\vec{B}}^\top \in \mathbb{R}^{(L_c R) \times d_v}. \tag{H.6}$$

We denote these solutions by

$$(\vec{W}, \vec{U}) = \texttt{tensorInv}(\vec{A}, \vec{B}, \widetilde{\vec{B}}).$$

Given an incoming chunk-boundary state $\mathbf{S}_{\text{in}} \in \mathbb{R}^{d_x \times d_v}$, define $\vec{Z} := \vec{U} + \vec{W}\mathbf{S}_{\text{in}} \in \mathbb{R}^{(L_c R) \times d_v}$. Then the chunk-exit state is

$$\mathbf{S}_{\text{out}} = \mathbf{S}_{\text{in}} + \vec{A}\vec{Z}, \tag{H.7}$$

which is the form used by Algorithm 3 (with the additional scalar decay γ_t handled there via chunk-local renormalization).

H.3 Mapping Falcon-3 to ParallelFlow

We now explicitly map the **Falcon-3** (Sliding Regression) update rule derived in Section 4 to this low-rank CDE framework.

The Falcon-3 Recurrence. Recall the mini-batch update with window size B (ignoring the scalar decay $\gamma_t = 1 - \eta_t \lambda_t$, which our chunked implementations handle via the same chunk-local log-decay renormalization used throughout):

$$\begin{aligned}\mathbf{S}_t &= \underbrace{\left(\mathbf{I} - \eta_t \bar{\mathbf{C}}_t^{(B)}\right)}_{\text{Transition Matrix}} \mathbf{S}_{t-1} + \underbrace{\eta_t \bar{\mathbf{N}}_t^{(B)}}_{\text{Input Injection}} \\ &= \left(\mathbf{I} - \frac{\eta_t}{B_t} \sum_{j \in \mathcal{I}_t} \mathbf{x}_j \mathbf{x}_j^\top\right) \mathbf{S}_{t-1} + \frac{\eta_t}{B_t} \sum_{j \in \mathcal{I}_t} \mathbf{x}_j \mathbf{v}_j^\top, \end{aligned} \tag{H.8}$$

where $B_t := |\mathcal{I}_t| \le B$, $\mathbf{x}_j := \phi(\mathbf{k}_{j-1})$ is the shifted write feature, $\bar{\mathbf{C}}_t^{(B)} := \mathbf{C}_t^{(B)}/B_t$, and $\bar{\mathbf{N}}_t^{(B)} := \mathbf{N}_t^{(B)}/B_t$. The unkernelized case has ϕ equal to the identity.

Identification of Drivers. To match the fixed-rank `tensorInv` kernel used in Algorithm 3, we zero-pad every active window to width B. Let $\mathcal{I}_t = \{j \mid \max(2, t-B+1) \le j \le t\}$, let $B_t := |\mathcal{I}_t| \le B$, and enumerate $\mathcal{I}_t$ increasingly as $(j_1, \ldots, j_{B_t})$. Define the padded write-feature and value blocks

$$\widetilde{\vec{X}}_t := \begin{bmatrix} \mathbf{x}_{j_1} & \mathbf{x}_{j_2} & \ldots & \mathbf{x}_{j_{B_t}} & \mathbf{0} & \ldots & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{d_x \times B}, \tag{H.9}$$

$$\widetilde{\vec{V}}_t := \begin{bmatrix} \mathbf{v}_{j_1} & \mathbf{v}_{j_2} & \ldots & \mathbf{v}_{j_{B_t}} & \mathbf{0} & \ldots & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{d_v \times B}, \tag{H.10}$$

where padded columns are zero. Then, for $t \ge 2$,

$$\bar{\mathbf{C}}_t^{(B)} = \frac{1}{B_t} \widetilde{\vec{X}}_t \widetilde{\vec{X}}_t^\top, \qquad \bar{\mathbf{N}}_t^{(B)} = \frac{1}{B_t} \widetilde{\vec{X}}_t \widetilde{\vec{V}}_t^\top,$$

since the padded columns vanish identically. Substituting into Eq. (H.8) yields the canonical left-multiplicative low-rank form

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \underbrace{\left(\frac{\eta_t}{B_t} \widetilde{\vec{X}}_t\right)}_{\vec{A}_t} \underbrace{\left(-\widetilde{\vec{X}}_t^\top\right)}_{\vec{B}_t^\top} \mathbf{S}_{t-1} + \underbrace{\left(\frac{\eta_t}{B_t} \widetilde{\vec{X}}_t\right)}_{\vec{A}_t} \underbrace{\widetilde{\vec{V}}_t^\top}_{\widetilde{\vec{B}}_t^\top}, \qquad (t \ge 2). \tag{H.11}$$

Thus the `tensorInv` kernel always uses the fixed rank parameter $R = B$ after padding:

$$\vec{A}_t = \frac{\eta_t}{B_t} \widetilde{\vec{X}}_t \in \mathbb{R}^{d_x \times B}, \qquad \vec{B}_t = -\widetilde{\vec{X}}_t \in \mathbb{R}^{d_x \times B}, \qquad \widetilde{\vec{B}}_t = \widetilde{\vec{V}}_t \in \mathbb{R}^{d_v \times B}.$$

For the boundary step $t = 1$, we set $\vec{A}_1 = \mathbf{0}$ so the update is a strict no-op. As in the main text, the stacked triangular solve uses the block-strict-causal mask over (time, rank) pairs, so same-time padded-window columns do not interact, and every residual is still evaluated at the pre-update state $\mathbf{S}_{t-1}$. In Algorithm 3, after the positive-decay renormalization of Eq. (4.16), one simply replaces η_t by $\widehat{\eta}_t$ and rescales the entire step-t value block by the common factor $(\delta_{i-1}^{(k)})^{-1}$ inside chunk k.

Complexity. Because the discrete Falcon-3 update is exactly a rank-B low-rank affine recurrence after padding (and may be viewed as a forward-Euler discretization of a corresponding CDE), we can utilize the `tensorInv` algorithm without approximation at the discrete level. For the small sliding windows used in our experiments (e.g., $B = 4$), the rank-B overhead is modest in practice and remains far cheaper than full $\mathcal{O}(L^2)$ attention while avoiding dense $d_x \times d_x$ state propagation. This derivation shows that Falcon-3 can use chunk-parallel scan methods while retaining the sliding-window regressor update exactly at the discrete level.