PRISM: Parallel Residual Iterative Sequence Model

Anonymous Authors[1]

Abstract

Generative sequence modeling faces a fundamental tension between the expressivity of Transformers and the efficiency of linear sequence models. Existing efficient architectures are theoretically bounded by shallow, single-step linear updates, while powerful iterative methods like Test-Time Training (TTT) break hardware parallelism due to state-dependent gradients. We propose PRISM (Parallel Residual Iterative Sequence Model) to resolve this tension. PRISM introduces a solver-inspired inductive bias that captures key structural properties of multi-step refinement in a parallelizable form. We employ a Write-Forget Decoupling strategy that isolates non-linearity within the injection operator. To bypass the serial dependency of explicit solvers, PRISM utilizes a two-stage proxy architecture: a short-convolution anchors the initial residual using local history energy, while a learned predictor estimates the refinement updates directly from the input. This design distills structural patterns associated with iterative correction into a parallelizable feedforward operator. Theoretically, we prove that this formulation achieves Rank-L accumulation, structurally expanding the update manifold beyond the single-step Rank-1 bottleneck. Empirically, it achieves comparable performance to explicit optimization methods while achieving **174x higher throughput**. Codes are available in anonymous.4open.science.

1. Introduction

Generative sequential modeling (e.g., HSTU(Zhai et al., 2024)) has revolutionized recommender systems by unlocking scaling laws for long-term user interests(Deng et al., 2025). However, the Transformer's quadratic complexity ($O(N^2)$) makes modeling lifelong sequences prohibitively expensive(Vaswani et al., 2017). Current solutions often resort to lossy heuristics like sliding windows or static compression to reduce costs, inevitably severing critical dependencies. Consequently, developing an efficient architecture capable of handling ultra-long sequences without sacrificing fidelity remains a key challenge.

Sequence modeling is fundamentally a problem of online memory optimization(Katharopoulos et al., 2020; Sun et al., 2024). While efficient architectures like Linear Attention(Yang et al., 2024b;a) and State Space Models (SSMs)(Gu et al., 2020; Gu & Dao, 2024) approximate the capabilities of Transformers with linear scaling, they are theoretically bounded by their reliance on a *One-Shot Optimization* paradigm. At each step, these models attempt to compress the semantic complexity of a new token into the memory state via a single geometric projection (equivalent to a single gradient descent step). This shallow update mechanism creates a *Linearity Bottleneck*: standard linear updates are structurally confined to Rank-1 modifications, limiting the model's ability to capture high-order feature correlations or perform the iterative refinement necessary to precisely minimize reconstruction error(Lei et al., 2025). Explicit optimization methods, such as Test-Time Training (TTT)(Schlag et al., 2021; Zhang et al., 2025; Behrouz et al., 2025b), resolve this by performing multi-step non-linear gradient descent at runtime. However, TTT falls into the *The Serial Dependency Bottleneck*: the gradients required for optimization are state-dependent, creating a sequential dependency chain that breaks the parallelism of modern hardware and negates the throughput advantages of efficient models(Yang et al., 2023; Behrouz et al., 2024).

To resolve the tension between expressivity and efficiency, we draw inspiration from the principles of numerical optimization, specifically iterative refinement. However, executing explicit optimization at runtime introduces prohibitive serial dependencies. Instead, we propose **PRISM** (**P**arallel **R**esidual **I**terative **S**equence **M**odel). We resolve the tension by introducing an amortized refinement mechanism inspired by iterative optimization, without claiming equivalence to explicit solvers. Our core hypothesis is that while the exact gradient trajectory is state-dependent, the optimal update structure—specifically Rank-L accumulation and Non-linear shaping—can be distilled into a learned static policy grounded in the input terrain. This allows the model

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

to emulate the functional benefits of deeper optimization steps as a structural inductive bias.

To operationalize this insight, we formulate PRISM around a Write-Forget Decoupling strategy. We maintain a hardware-efficient linear decay for the forgetting dynamics while isolating the computational complexity within the injection operator. Crucially, to bypass the serial bottleneck of iterative solvers, we introduce an Input-Anchored Loop Unrolling architecture. This design employs a two-stage proxy mechanism: it first anchors the initial residual via a short-convolution that captures local history energy, and then applies a learned predictor to estimate the multi-step refinement updates. This allows us to "collapse" the temporal complexity of a non-linear optimization loop into a single, fused parallel operator.

Our main contributions are:

- **Amortized Residual Optimization Framework:** We reframe sequence modeling as a residual fitting problem and propose Write-Forget Decoupling. This theoretical framework justifies fixing the decay dynamics to maintain stability while allocating model capacity to a Rank-L, non-linear injection operator that approximates multi-step gradient boosting.

- **PRISM Architecture:** We propose a hardware-aware implementation that employs a two-stage proxy mechanism—initial anchoring via short convolutional operations followed by refinement through a learned predictor—to circumvent the inherent serial computation bottleneck. Empirically, PRISM attains performance comparable to that of explicit iterative solvers, demonstrating modeling capacity on par with deep Transformer architectures on challenging benchmarks, while simultaneously achieving throughput improvements of up to **174x** relative to explicit optimization-based methods.

- **Theoretical Expressivity:** We formally prove that our input-anchored formulation yields structural properties of *Rank Accumulation*. This demonstrates that PRISM structurally expands the hypothesis space beyond single-step updates, asymptotically approaching the solution of an ideal high-order non-linear delta rule.

2. Related Work

We survey efficient sequence modeling through the lens of **Optimization Fidelity**, distinguishing between architectures that rely on fixed heuristic updates versus those that approximate explicit optimization processes. A comparison is provided in Table 1.

2.1. Linear Recurrences and State Space Models

The dominant approach for efficient modeling is to constrain the interaction to a linear recurrence $\mathbf{S}_t = \mathbf{A}_t\mathbf{S}_{t-1} + \mathbf{B}_t\mathbf{x}_t$. We categorize these based on how they construct the recurrence operators.

Additive and Gated Updates (RWKV, Mamba). Early efficient transformers (Katharopoulos et al., 2020) and RWKV (Peng et al., 2025) utilize a Hebbian update rule $\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t\mathbf{k}_t^\top$, which purely accumulates associations without explicitly correcting errors. State Space Models (SSMs) like Mamba (Gu & Dao, 2024) and Mamba2 (Dao & Gu, 2024) advance this by introducing **Input-Dependent Decay** via discretization parameters $(\Delta, \mathbf{A}, \mathbf{B})$. While Mamba's selection mechanism significantly improves context filtering, its injection operator remains a Rank-1 outer product ($\mathbf{B}_t\mathbf{C}_t^\top$ in SSM terms).

2.2. Sequence Modeling as Online Optimization

A more theoretically grounded perspective views sequence modeling as an online optimization problem, where the goal is to minimize a reconstruction loss $\mathcal{L}_t = \|\mathbf{v}_t - \mathbf{S}_{t-1}\mathbf{k}_t\|^2$.

Linear Delta Rules (DeltaNet, RetNet). DeltaNet (Yang et al., 2024b) explicitly formulates the update as a single step of gradient descent:

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

where β_t acts as a learning rate. RetNet (Sun et al., 2023) employs a similar residual structure combined with exponential decay. While these methods introduce an "error correction" term, they are structurally constrained to be **Linear** to satisfy the associativity required for parallel prefix scans. Consequently, they effectively perform "blind" optimization: they cannot adjust the update direction based on the current memory landscape (state-dependent gain) without breaking parallelism.

Test-Time Training (TTT). To overcome the expressivity bottleneck of linear rules, TTT (Sun et al., 2024) proposes performing explicit, multi-step gradient descent on the hidden state during inference. This allows for non-linear updates (in TTT-NN) that adaptively reshape the memory. However, as analyzed in *Atlas*, this approach faces the **Seriality Trap**: calculating the exact gradient $\nabla_{\mathbf{S}_{t-1}}\mathcal{L}$ introduces a strict dependency on the previous state $\mathbf{S}_{t-1}$. This precludes the use of parallel scan algorithms, forcing the model to run sequentially during training, which is prohibitively slow for large-scale pretraining.

2.3. Positioning PRISM: Amortized Optimization

PRISM resolves the tension between the *parallel efficiency* of linear models and the *optimization fidelity* of TTT. Unlike DeltaNet which relies on a shallow Rank-1 update,

Table 1. Comparison of sequence modeling paradigms. PRISM bridges the gap between efficient Linear Recurrences and expressive Optimization-based models. Unlike TTT which hits a serial barrier due to state-dependent gradients, PRISM employs an **Input-Anchored** proxy to amortize the high-rank refinement cost, enabling fully parallel training.

Paradigm	Representative Models	Update Rule	Rank	Optimization Type	Parallelism
Linear Attention	Linear Trans., RWKV	Additive (Hebbian)	1	Heuristic	✓
State Space Models	Mamba, Mamba2	Gated Decay	1	Discretized ODE	✓
Linear Delta	DeltaNet, RetNet	Residual Error	1	One-Step (Linear)	✓
Test-Time Training	TTT-Linear/NN	Gradient Descent	Full	Multi-Step (Exact)	✗
PRISM (Ours)	**-**	**Amortized Residual**	L	**Multi-Step (Proxy)**	✓

and unlike TTT which requires serial state access, PRISM introduces **Input-Anchored Loop Unrolling**. By approximating the state-dependent terms (e.g., the residual error and contextual gain) using a local convolutional proxy, PRISM constructs a **High-Rank** ($L > 1$) update operator that can be computed in parallel. This allows PRISM to emulate the trajectory of a multi-step iterative solver while retaining the $O(N)$ training throughput of standard linear attention.

3. Theoretical Motivation

In this section, we reframe linear attention through the lens of numerical optimization to identify its expressivity bottleneck. We then introduce an ideal nonlinear update rule that restores adaptive, context-aware dynamics, and formally analyze why its state-dependent structure precludes parallel computation.

3.1. The Optimization Gap: From Blind Updates to Ideal Solvers

Linear Attention as Online Learning. Let $\mathbf{k}_t \in \mathbb{R}^d$ and $\mathbf{v}_t \in \mathbb{R}^d$ denote the key and value vectors at timestep t, where d is the dimension of the key and value space. Linear attention avoids the quadratic cost of standard attention by maintaining a compact state matrix $\mathbf{S}_t$ that summarizes past keys and values associations. Specifically, standard linear attention updates its state via

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top. \tag{2}$$

We view this not as a heuristic, but as a single step of online gradient descent on a linear objective

$$\mathcal{L}_t(\mathbf{S}) = -\langle \mathbf{S}\mathbf{k}_t, \mathbf{v}_t \rangle, \tag{3}$$

which measures how well the projected key $\mathbf{S}\mathbf{k}_t$ aligns with the value $\mathbf{v}_t$. Standard linear attention updates $\mathbf{S}$ via online gradient descent with fixed learning rate β :

$$\mathbf{S}_t = \mathbf{S}_{t-1} - \beta \nabla_{\mathbf{S}} \mathcal{L}_t(\mathbf{S}_{t-1}) = \mathbf{S}_{t-1} + \beta \mathbf{v}_t \mathbf{k}_t^\top, \tag{4}$$

where $\beta = 1$ in conventional formulations.

The Limitation. This update is structurally rigid in two critical aspects. First, it applies a rank-1 correction $\eta \mathbf{v}_t \mathbf{k}_t^\top$ at every step, regardless of the current state $\mathbf{S}_{t-1}$. Second, it lacks the capacity to correct errors or model complex dependencies where the update magnitude should depend on the current memory saturation. Appendix B further analyzes the issues caused by the absence of nonlinearity.

The Ideal Non-Linear Rule. To transcend the rigidity of linear updates, one can consider a non-linear prediction model in which the value $\mathbf{v}_t$ is predicted by $\sigma(\mathbf{S}\mathbf{k}_t)$, where $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ is a non-linear function. This leads to the following local objective:

$$\mathcal{L}_t(\mathbf{S}) = \frac{1}{2} \|\sigma(\mathbf{S}\mathbf{k}_t) - \mathbf{v}_t\|_2^2. \tag{5}$$

We refer to the update rule obtained by minimizing this objective as the gradient-based *Ideal Solver* (detailed in Appendix C):

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \sigma'(\mathbf{S}_{t-1}\mathbf{k}_t) \odot (\mathbf{v}_t - \sigma(\mathbf{S}_{t-1}\mathbf{k}_t)) \cdot \mathbf{k}_t^\top, \tag{6}$$

where the contextual gain is given by $\sigma'(\mathbf{S}_{t-1}\mathbf{k}_t)$ and $\odot$ is element-wise product. Crucially, this gain term introduces state-dependent modulation: in saturated regimes, where $\sigma(\mathbf{S}_{t-1}\mathbf{k}_t)$ is near its asymptote and σ' is small, the update is automatically suppressed to prevent overshooting; conversely, in sensitive regions where the memory response is still in the linear range of σ, the gain is large, enabling rapid adaptation to new information. Thus, the Ideal Solver not only corrects errors but also adapts its learning behavior to the current memory state, offering a principled path toward expressive and stable long-sequence modeling.

3.2. The Parallelism Conflict

Modern linear attention leverages *parallel prefix scans* (e.g., the Blelloch algorithm) to compute recurrent states in $O(\log N)$ time. This requires the recurrence to be expressed in a state-independent form:

$$\mathbf{S}_t = \mathbf{S}_{t-1}\mathbf{A}_t + \mathbf{B}_t, \tag{7}$$

where the transition operators $\mathbf{A}_t$ and $\mathbf{B}_t$ should be computable *solely from the input sequence up to step* t , without

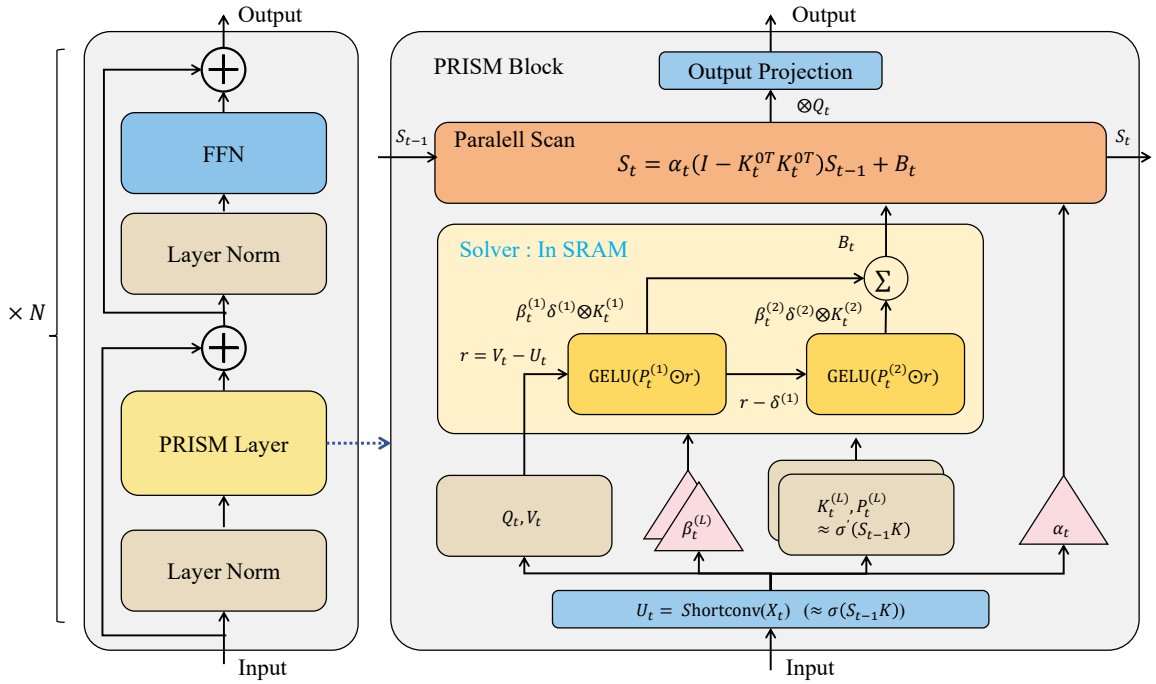

Figure 1. **The PRISM Architecture.** The framework operates in two phases to approximate the Ideal Non-Linear Solver within a parallelizable linear recurrence. **Phase 1 (Input-Anchored Simulation):** A ShortConv anchor captures the local pre-activation proxy ($u_t \approx S_{t-1}k_t$). Parallel predictors (p) generate the *Contextual Gain* vectors to simulate the derivative of the activation (σ'), while basis projections (K) determine the geometric subspaces. **Phase 2 (Iterative Rank Accumulation):** An unrolled residual loop constructs the high-rank injection matrix $\mathbf{B}$. Each layer l performs a *Greedy Residual Subtraction*, adding an orthogonal rank-1 update to the accumulator. This expands the update manifold from Rank-1 to Rank-L. **State Update:** The accumulated high-rank update is injected into a *Decoupled Linear Recurrence*, where the forgetting operator $\mathbf{A}$ remains state-independent to preserve parallel scan efficiency.

dependence on the hidden state $\mathbf{S}_{t-1}$. We formalize this requirement as follows:

Definition 3.1 (**State-Independent Recurrence**). A linear recurrence $\mathbf{S}_t = \mathbf{S}_{t-1}\mathbf{A}_t + \mathbf{B}_t$ admits parallel prefix scanning if and only if there exist functions f_A and f_B such that for all t ,

$$\mathbf{A}_t = f_A(\mathbf{x}_{\leq t}), \quad \mathbf{B}_t = f_B(\mathbf{x}_{\leq t}), \tag{8}$$

i.e., $\mathbf{A}_t$ and $\mathbf{B}_t$ are independent of $\mathbf{S}_{t-1}$.

While the ideal solver provides necessary expressivity, it fundamentally conflicts with the parallelization strategy that enables efficient long-sequence modeling in linear attention, as its injection term depends recursively on the hidden state through both the contextual gain $\sigma'(\mathbf{S}_{t-1}\mathbf{k}_t)$ and the nonlinear prediction $\sigma(\mathbf{S}_{t-1}\mathbf{k}_t)$. This locks the computation into a serial chain ($O(N)$ latency), prohibiting parallel prefix scans (detailed in Appendix C).

4. Method: The PRISM Architecture

4.1. The Write-Forget Decoupling Hypothesis

To resolve the conflict between the ideal solver's non-linear expressivity and the requirement for hardware parallelism (Definition 3.1), we ask: *which component of the recurrence can be safely linearized without catastrophic loss of performance?* In Appendix D, we answer this through a rigorous spectral perturbation analysis, modeling the sequence as a discretized ODE via Zero-Order Hold. Our findings reveal a fundamental asymmetry in error propagation:

- **Logarithmic Stability of Forgetting ($\mathbf{A}_t$):** We prove that for stable systems, approximation errors in the multiplicative forgetting operator accumulate at a sub-linear rate—specifically, $O(\ln T)$ in the worst case and $O(1)$ on average (Theorem D.2 & D.3). This implies that $\mathbf{A}_t$ is structurally robust and can be safely approximated by a structured, low-rank proxy (e.g., the Gated DeltaNet parameterization $\mathbf{I} - \beta\mathbf{k}\mathbf{k}^\top$) without degrading long-term memory (Gu et al., 2020; Yang et al., 2024a).
- **Linear Divergence of Injection ($\mathbf{B}_t$):** Conversely, we demonstrate that additive errors in the injection term accumulate linearly ($O(T)$) in persistent memory channels. This indicates that the system is hypersensitive to the rank-fidelity of $\mathbf{B}_t$, forcing us to retain a high-rank, non-linear estimator to prevent irreversible semantic information loss.

Based on this theoretical dichotomy, we propose **Write-**

Forget Decoupling. We restrict the forgetting dynamics ($\mathbf{A}_t$) to a strict Rank-1 update for efficiency, while allocating the majority of the model's expressivity to a parallel, iterative solver that constructs a High-Rank injection matrix ($\mathbf{B}_t$).

4.2. Amortized Input-Anchored Simulation

Now we introduce the detailed design of $\mathbf{B}_t$. To implement the non-linear injection $\mathbf{B}_t$ in parallel, we bypass the serial dependency on $\mathbf{S}_{t-1}$ via **Input-Anchored Simulation**. The key insight is that, for many sequential tasks, the interaction $\mathbf{S}_{t-1}\mathbf{k}_t$, which governs the contextual information, is dominated by recent history. This motivates us to approximate optimization trajectory it using only the local input context, without full state recurrence. Concretely, we recover a proxy of the state interaction $\mathbf{u}_t \approx \mathbf{S}_{t-1}\mathbf{k}_t$ using a short convolution:

$$\mathbf{u}_t = \text{ShortConv}(\mathbf{X}_{\leq t}) \approx \mathbf{S}_{t-1}\mathbf{k}_t. \tag{9}$$

This captures the high-frequency signal dominated by local context and serves as the anchor for all subsequent approximations. We provide a theoretical analysis of the approximation bias of this local proxy in Appendix E, showing that the error decays exponentially with the effective memory window under fading-memory dynamics.

As discussed before, $\mathbf{u}_t$ enables state-independent access to the key interaction, but a single rank-1 update is insufficient to capture the full expressivity of the ideal solver. Hence, we employ iterative rank accumulation, a multi-step refinement process that constructs $\mathbf{B}_t$ as a sum of L orthogonal rank-1 components(detailed in Appendix F and G):

$$\mathbf{B}_t = \sum_{l=1}^{L} \beta_t^{(l)} \cdot (\boldsymbol{\delta}_t^{(l)} \otimes \mathbf{k}_t^{(l)}), \tag{10}$$

where the scalar gate $\beta_t^{(l)}$ acts as a confidence score, scaling the contribution of each component. Notably, each gate value is computed via the anchor

$$\beta_t^{(l)} = \mathbf{W}_{beta}^{(l)}\mathbf{u}_t. \tag{11}$$

We explicitly align the structure of $\boldsymbol{\delta}_t^{(l)}$ and $\mathbf{k}_t^{(l)}$ with the corresponding terms in (6), ensuring that the parallelized update preserves its essential geometric properties. More specifically, we project the anchor to obtain the geometric basis $\mathbf{k}_t^{(l)}$ for each component:

$$\mathbf{k}_t^{(l)} = \mathbf{W}_k^{(l)}\mathbf{u}_t. \tag{12}$$

The update direction $\boldsymbol{\delta}_t^{(l)}$ is formed by modulating a residual estimate with a simulated contextual gain—both derived from the anchor $\mathbf{u}_t$. We first train a multi-layer predictor p to simulate the contextual gain $\sigma'(\mathbf{S}_{t-1}\mathbf{k}_t)$, i.e.,

$$p_t^{(l)} = \mathbf{W}_p^{(l)}\mathbf{u}_t \approx \sigma'(\mathbf{S}_{t-1}\mathbf{k}_t). \tag{13}$$

For residual term $\mathbf{v}_t - \sigma(\mathbf{S}_{t-1}\mathbf{k}_t)$, we estimate the initial error signal $\mathbf{r}_t^{(1)}$ by comparing the target $\mathbf{v}_t$ against the local proxy, rather than the true state:

$$\mathbf{r}_t^{(1)} = \mathbf{v}_t - \mathbf{u}_t \approx \mathbf{v}_t - \sigma(\mathbf{S}_{t-1}\mathbf{k}_t). \tag{14}$$

This initial residual seeds an iterative refinement process. At each layer $1 \leq l \leq L$, we compute an update direction $\boldsymbol{\delta}_t^{(l)}$ by modulating $\mathbf{r}_t^{(l)}$ with the simulated gain $p_t^{(l)}$, serveing as a localized newton step, and then subtract this correction to form the next residual:

$$\boldsymbol{\delta}_t^{(l)} = \text{GELU}\big(p_t^{(l)} \odot \mathbf{r}_t^{(l)}\big), \tag{15}$$

$$\mathbf{r}_t^{(l+1)} = \mathbf{r}_t^{(l)} - \boldsymbol{\delta}_t^{(l)}. \tag{16}$$

This orthogonal subtraction encourages subsequent layers to capture complementary error signals, thereby enriching the information content of the total injection.

Following the decoupling strategy, the forgetting operator takes the linear form

$$\mathbf{A}_t = \mathbf{I} - \beta_t^{(1)}\mathbf{k}_t^{(1)} \otimes \mathbf{k}_t^{(1)}, \tag{17}$$

reusing the first-layer projection $\mathbf{k}_t^{(1)}$ to ensure both simplicity and alignment with the write path.

Finally, the accumulated matrix $\mathbf{B}_t$ is injected into the recurrent state.

$$\mathbf{S}_t = \alpha_t\mathbf{S}_{t-1}(\mathbf{I} - \beta_t^{(1)}\mathbf{k}_t^{(1)} \otimes \mathbf{k}_t^{(1)}) + \sum_{l=1}^{L} \beta_t^{(l)} \cdot (\boldsymbol{\delta}_t^{(l)} \otimes \mathbf{k}_t^{(l)}). \tag{18}$$

5. Experiments

Our experiments aim to answer three key research questions:

- **RQ1 (Effectiveness):** Does PRISM achieve the modeling fidelity of explicit iterative solvers (e.g., TTT) and deep Transformers via amortized optimization?
- **RQ2 (Scalability):** How does PRISM perform on long-tail and ultra-long sequences compared to existing linear baselines, and does it maintain $O(N)$ efficiency?
- **RQ3 (Mechanism):** Do the Input-Anchored Solver components (ShortConv Proxy and Unrolled Loop) function as hypothesized to enable rank-L updates?

5.1. Experimental Setup

Datasets. We evaluate on four widely-used recommendation benchmarks: *Amazon Books*, *Amazon Movies*, *Amazon Elecs*, and *Yelp*(detailed in Appendix I). Recommendation is selected as a stress test for High-Rank Sparsity, as user interests are inherently multi-modal and diverse, contrasting with the low-rank attention often found in NLP.

Baselines. We compare PRISM against a comprehensive suite of state-of-the-art sequence models, categorized into three groups: (1) **Transformers (Upper Bound):** We include SASRec (Kang & McAuley, 2018) and HSTU (Zhai et al., 2024) as strong non-linear baselines to estimate the theoretical performance ceiling; (2) **Linear Recurrences (Decay-Based):** We evaluate heuristic linear models including SLA (Katharopoulos et al., 2020), GLA (Yang et al., 2024a), GSA (Zhang et al., 2024), MoM (Du et al., 2025), and the state-space model Mamba-2 (Dao & Gu, 2024); (3) **Optimization-Based Models:** We compare against gradient-inspired architectures including Gated DeltaNet (Yang et al., 2024a), TTT-Linear (Sun et al., 2024) ATLAS(Behrouz et al., 2025a), and TITANS (Behrouz et al., 2024).(detailed in Appendix J)

Metrics. We report NDCG@10 and Hit@10. To assess efficiency, we measure Training Throughput (thousand tokens/s) on an NVIDIA H20 GPU.

5.2. Main Recommendation Performance (RQ1)

Table 2 and Table 8 summarizes the performance on recommendation benchmarks.

- **Dominance of Non-Linear Optimization.** A clear trend is observed: models employing optimization-inspired or non-linear update mechanisms (TITANS, PRISM, ATLAS) consistently outperform standard heuristic Linear Attention models (SLA, GLA, MoM). For instance, the top three linear models in Mean Rank all belong to this category, whereas purely associative models generally rank lower. This empirical evidence strongly validates the advantage of introducing **High-Rank** or **Non-Linear** structures into the state update rule to capture complex user interest evolution.

- **Amortized Efficiency with Explicit Quality.** PRISM achieves performance parity with (and in some cases surpasses) explicit optimization models like TTT and TITANS. Notably, on *Amazon Movies*, PRISM achieves the highest AUC among all linear models, outperforming even the sophisticated TITANS. This confirms that our *Input-Anchored Proxy* successfully distills the trajectory of explicit solvers into a feed-forward operator, without the need for actual gradient descent at test time.

- **Closing the Gap with Transformers.** While $O(N^2)$ Transformers (SASRec, HSTU) still serve as the performance ceiling due to their full context visibility, the gap between PRISM and these quadratic models is surprisingly narrow. On *Amazon Books*, PRISM's AUC is virtually identical to SASRec. This suggests that the "expressivity tax" of linearizing attention is becoming negligible with advanced state-tracking mechanisms like PRISM.

- **Efficiency Preview.** It is crucial to note that while TTT and TITANS achieve high accuracy, they suffer from serial bottlenecks due to state-dependent gradient calculations. In contrast, PRISM maintains fully parallelizable dynamics. As we will demonstrate in the following efficiency analysis (Section 5.3), PRISM outperforms these optimization-based baselines and Transformers by orders of magnitude in training throughput and inference latency.

5.3. Training Efficiency

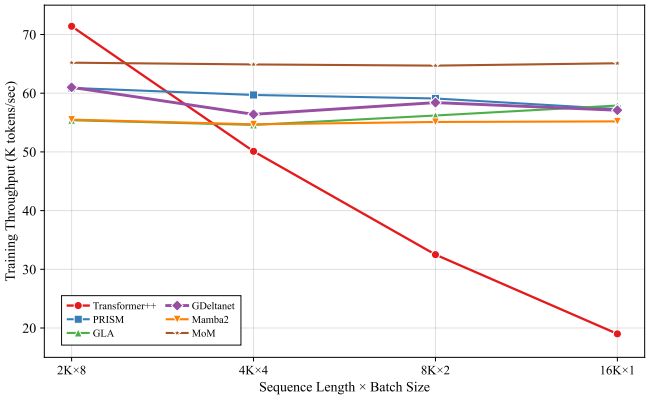

Figure 2. Training throughput comparison of 0.13B models on a single H20 GPU.

A key advantage of PRISM is its robustness to sequence length and its hardware-aware design. We analyze the computational benefits of our Input-Anchored parallelization strategy. Figure 2 shows the training throughput as a function of sequence length and batch size for different 0.13B models on a single H20 GPU, where all recurrent models adopt the materialization version of Flash-Linear-Attention(Yang & Zhang, 2024). Transformer++ achieves the highest throughput at short sequences (71.4K tokens/s at 2K) thanks to the highly optimized Flash-Attention-2 kernel(Dao, 2023), but suffers from severe degradation (3.8× slowdown) as sequence length increases to 16K due to quadratic complexity.

In contrast, PRISM—implemented using the operators in Flash-Linear-Attention maintains stable throughput (∼

Table 2. Recommendation benchmark performance. Metrics are Hit@200 and NDCG@200 with per-dataset AUC. Best linear-attention results are in **bold** and second-best are underlined.

Model	Amazon_books			Amazon_movies			Amazon_elec			Yelp			Mean Rank
	H@200	NDCG@200	AUC	H@200	NDCG@200	AUC	H@200	NDCG@200	AUC	H@200	NDCG@200	AUC	
Linear Attention													
SLA	0.1129	0.0212	0.8866	0.1137	0.0227	0.7461	0.1290	**0.0243**	0.7023	0.1627	0.0311	0.9392	6.75
GLA	0.0879	0.0158	0.8752	0.1193	0.0222	0.7478	0.1196	0.0210	0.7008	0.1129	0.0220	0.8943	9.38
MoM	0.0854	0.0158	0.8705	0.1397	0.0281	0.7705	0.1333	0.0238	0.7042	0.1642	0.0306	0.9346	5.25
GSA	0.1226	0.0233	0.8870	0.1160	0.0222	0.7427	0.1318	0.0228	0.7087	0.1629	0.0307	0.9383	7.00
MAMBA2	0.1234	0.0234	0.8872	0.1372	0.0276	0.7713	0.1338	0.0237	0.7157	0.1621	0.0306	0.9385	5.00
ATLAS	0.1190	0.0223	0.8884	0.1367	0.0278	0.7710	**0.1421**	**0.0243**	0.7042	0.1629	0.0304	0.9383	5.00
GDeltanet	0.1214	0.0230	0.8844	0.1241	0.0253	0.7504	0.1333	0.0242	**0.7159**	0.1648	0.0308	0.9367	5.12
TTT	0.1255	0.0234	0.8871	0.1288	0.0256	0.7591	0.1344	0.0234	0.6946	0.1636	0.0306	0.9375	5.25
TITANS	**0.1272**	**0.0243**	0.8869	0.1358	0.0278	0.7652	0.1313	0.0233	0.7007	**0.1653**	**0.0313**	**0.9395**	3.62
PRISM	0.1258	0.0238	**0.8888**	**0.1411**	**0.0289**	**0.7727**	0.1409	0.0237	0.7134	0.1637	0.0310	0.9393	**2.62**
Transformer													
SASRec	0.1138	0.0215	0.8910	0.1272	0.0244	0.7677	0.1438	0.0273	0.7293	0.1723	0.0325	0.9410	–
HSTU	0.1224	0.0233	0.8835	0.1399	0.0293	0.7748	0.1407	0.0250	0.7189	0.1595	0.0295	0.9324	–

$61K \rightarrow$57K tokens/s) across all context lengths, performing on par with Gated DeltaNet, GLA, Mamba2, and MoM (all operating within the 55–65K tokens/s range). Notably, TTT(0.34K tokens/s) is over **170× slower than PRISM** due to its sequential computation nature, underscoring the critical importance of hardware-efficient algorithms for practical training scalability.

5.4. Ablation Study: Deconstructing the Solver (RQ3)

To verify the solver design and quantify the contribution of each component, we conduct a component analysis on the *Amazon Electronics* dataset. We compare the full PRISM model against four variants:

- **w/o Iterative Refinement ($L = 1$):** Reduces the solver depth to a single step, mimicking standard DeltaNet-like updates.
- **w/o Non-Linearity:** Removes the GELU activation inside the solver, enforcing a strictly linear update.
- **w/o ShortConv Anchor:** Removes the local convolution, driving the update solely by the current token projection.
- **w/o Gain Predictor:** Removes the learned gain branch (p-projection), utilizing a constant step size.

Table 3 summarizes the results. We observe that the full PRISM model consistently outperforms all ablated variants across both Hit Rate and AUC metrics, confirming that every component plays an essential role in the system.

Impact of Iterative Depth. The most substantial performance degradation occurs when removing the iterative refinement (reducing the solver to $L = 1$). This represents the performance floor among all variants, strongly supporting our core hypothesis: a single-step (Rank-1) update is structurally insufficient for capturing complex user interest transitions. The "depth" of the solver is therefore the most critical factor for expanding the model's memory capacity.

Table 3. Ablation study on *Amazon Electronics*. We report Hit@K (H@K) at $K = \{200, 500\}$ and AUC. Removing any component leads to a degradation in global ranking quality.

Model Variant	Hit@K		AUC
	200	500	
PRISM (Full)	**0.1409**	**0.2613**	**0.7134**
w/o Iterative Refinement ($L = 1$)	0.1155	0.2374	0.6805
w/o Non-Linearity	0.1316	0.2477	0.7047
w/o ShortConv Anchor	0.1406	0.2584	0.7076
w/o Gain Predictor	0.1383	0.2406	0.7098

Impact of Non-Linearity. Enforcing a strictly linear update (w/o Non-Linearity) leads to a clear decline in ranking quality. This validates that the "neural" nature of our solver—specifically the non-linear activation within the refinement loop—provides necessary expressivity, allowing the model to approximate complex optimization trajectories that standard linear matrix associations cannot capture.

Impact of Anchoring and Gating. Ablating the local ShortConv anchor or the learned gain predictor also results in suboptimal performance. The degradation in AUC across these variants suggests that without input-anchored local context or adaptive step sizes, the solver struggles to maintain a stable global ranking. This highlights that the optimization trajectory must be grounded in local feature context to be effective over long sequences.

5.5. Mechanistic Probing: Disentangling Storage from Computation

While our primary results establish PRISM's effectiveness on standard benchmarks, we additionally perform a controlled study to disentangle the underlying causes of these

performance gains (see Appendix H for details). In particular, we examine whether PRISM's advantage arises predominantly from increased information retention (Capacity) or from qualitatively different computational procedures (Reasoning).

Experimental Setup. We operate in a highly constrained resource-starved regime to force architectural bottlenecks to surface. All models are restricted to a tiny hidden dimension $D = 16$, vocabulary $V = 64$, and sequence length $N = 128$. We train for 10k steps.

- **PRISM:** Configured with refinement depth $L = 2$.
- **Linear Attention (LA)(Katharopoulos et al., 2020):** Represents the **Rank-1 Bottleneck** ($\Delta S = vk^\top$).
- **Mixture-of-Memories (MoM)(Du et al., 2025):** Configured with 4 experts. This represents **Spatial Rank Expansion**. By routing tokens to different linear heads, MoM increases capacity width-wise, but each expert remains internally linear.
- **Transformer(Vaswani et al., 2017):** Represents **Full-Rank Expressivity** (the upper bound).

Results & Analysis. Table 4 presents the accuracy across three task categories. We observe a distinct capability hierarchy:

Table 4. Mechanistic Probing Accuracy ($D = 16$). **Logic** and **Structure** tasks reveal the Linearity Wall, where LA and MoM fail (random chance) while PRISM succeeds, matching or exceeding the Transformer.

Category	Task	Trans.	LA	MoM	PRISM
Memory	MQAR	**0.98**	**0.98**	**0.98**	**0.98**
	Poly-Recall	**1.00**	**1.00**	**1.00**	**1.00**
	Var. Tracking	**0.40**	0.34	0.35	<u>0.37</u>
Logic	Parity Check	**1.00**	0.49	0.50	**1.00**
	Local XOR	**1.00**	0.50	0.49	**1.00**
Structure	Modulo Add	<u>0.23</u>	0.07	0.06	**0.50**
	Palindrome	0.49	0.49	0.50	**0.99**
Control	MUX Logic	**0.98**	**0.98**	0.97	**0.98**
	Silence Gate	**0.82**	0.51	0.50	<u>0.66</u>

1. The Memory Plateau (Storage). On pure associative tasks (MQAR, Poly-Recall), all models achieve approx. 100% accuracy. This confirms that for storage, the bottleneck is the physical state size ($d \times d$), not the update rule. PRISM's complex writing mechanism preserves the full storage efficiency of linear models.

2. The Linearity Wall (Logic). On XOR and Parity, a sharp divergence occurs.

- **LA & MoM Fail ($\approx 50\%$):** Crucially, MoM's failure reveals that *spatial parallelization* of linear experts does not yield *non-linear reasoning*. MoM fits multiple linear hyperplanes but cannot model the *nested* interaction required for parity.
- **PRISM Succeeds (100%):** By unrolling the solver ($L = 2$), PRISM performs iterative non-linear correction, effectively approximating the logical decision boundary within the recurrent update.

3. Structural Inductive Bias. Surprisingly, on **Palindrome** and **Modulo Add**, PRISM significantly outperforms even the Transformer in this constrained regime (e.g., Palindrome: 0.99 vs. 0.49). We attribute this to PRISM's **Input-Anchored** design: the ShortConv anchor provides a strong inductive bias for local structural features, whereas the Transformer struggles to learn precise positional arithmetic from scratch with limited heads and dimensions.

4. Gating Robustness. On **Silence Gate**, PRISM (0.66) outperforms linear baselines (0.50), demonstrating that its non-linear $\mathcal{P}(x)$ predictor can suppress noise more effectively than linear gates, though it still lags behind the Transformer's full attention (0.82).

6. Conclusion

In this work, we identify the **"One-Shot Bottleneck"** as the primary limitation of efficient sequence models. We argue that standard linear updates, while efficient, lack the optimization depth required to capture complex semantic dependencies. To bridge this gap without incurring quadratic costs or serial latencies, the state update mechanism must evolve from simple geometric projections to **Amortized Residual Optimization**. We introduce **PRISM**, a framework that resolves the tension between high-fidelity optimization and hardware efficiency. By employing a strategy of **Write-Forget Decoupling**, we maintain a stable linear decay while allocating computational capacity to a rank-L, non-linear injection operator. Crucially, we bypass the serial dependency of iterative solvers via **Input-Anchored Loop Unrolling**: utilizing a ShortConv Proxy to anchor the initial residual and a learned predictor to estimate the multi-step refinement trajectory. This innovation allows us to "collapse" a complex, non-linear optimization loop into a single, fused parallel operator. Our theoretical analysis proves that this unrolled design yields a hypothesis space strictly richer than standard linear attention, unlocking **Rank Accumulation** within a linear recurrence. Empirically, PRISM validates this design philosophy: it matches the modeling fidelity of explicit serial solvers while achieving **174x higher throughput**. These results suggest that **distilling** iterative optimization trajectories into amortized, input-anchored operators is a robust pathway for scaling efficient foundation models.

Impact Statement

This paper presents work whose goal is to advance the field of efficient sequence modeling by bridging the gap between optimization fidelity and hardware-efficient parallel training. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Behrouz, A., Zhong, P., and Mirrokni, V. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Behrouz, A., Li, Z., Kacham, P., Daliri, M., Deng, Y., Zhong, P., Razaviyayn, M., and Mirrokni, V. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025a.

Behrouz, A., Razaviyayn, M., Zhong, P., and Mirrokni, V. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization. *arXiv preprint arXiv:2504.13173*, 2025b.

Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

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

Deng, J., Wang, S., Cai, K., Ren, L., Hu, Q., Ding, W., Luo, Q., and Zhou, G. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965*, 2025.

Du, J., Sun, W., Lan, D., Hu, J., and Cheng, Y. Mom: Linear sequence modeling with mixture-of-memories. *arXiv preprint arXiv:2502.13685*, 2025.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Ré, C. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33: 1474–1487, 2020.

Kang, W.-C. and McAuley, J. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pp. 197–206. IEEE, 2018.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Lei, J., Zhang, D., and Poria, S. Error-free linear attention is a free lunch: Exact solution from continuous-time dynamics. *arXiv preprint arXiv:2512.12602*, 2025.

Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., Dalmedigos, I., Safahi, E., Meirom, S., Belinkov, Y., Shalev-Shwartz, S., et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Peng, B., Zhang, R., Goldstein, D., Alcaide, E., Du, X., Hou, H., Lin, J., Liu, J., Lu, J., Merrill, W., et al. Rwkv-7" goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

Schlag, I., Irie, K., and Schmidhuber, J. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pp. 9355–9366. PMLR, 2021.

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Sun, Y., Li, X., Dalal, K., Xu, J., Vikram, A., Zhang, G., Dubois, Y., Chen, X., Wang, X., Koyejo, S., et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Team, K., Zhang, Y., Lin, Z., Yao, X., Hu, J., Meng, F., Liu, C., Men, X., Yang, S., Li, Z., et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yang, S. and Zhang, Y. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024. URL https://github.com/fla-org/flash-linear-attention.

Yang, S., Wang, B., Shen, Y., Panda, R., and Kim, Y. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

Yang, S., Kautz, J., and Hatamizadeh, A. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024a.

Yang, S., Wang, B., Zhang, Y., Shen, Y., and Kim, Y. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024b.

Zhai, J., Liao, L., Liu, X., Wang, Y., Li, R., Cao, X., Gao, L., Gong, Z., Gu, F., He, M., et al. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. *arXiv preprint arXiv:2402.17152*, 2024.

Zhang, T., Bi, S., Hong, Y., Zhang, K., Luan, F., Yang, S., Sunkavalli, K., Freeman, W. T., and Tan, H. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025.

Zhang, Y., Yang, S., Zhu, R.-J., Zhang, Y., Cui, L., Wang, Y., Wang, B., Shi, F., Wang, B., Bi, W., et al. Gated slot attention for efficient linear-time sequence modeling. *Advances in Neural Information Processing Systems*, 37: 116870–116898, 2024.

A. Nomenclature and Definitions

To ensure clarity and consistency across the main text and appendices, we summarize the key mathematical symbols used in this paper in Table 5.

Table 5. Table of Notations.

Symbol	Description
L	Number of refinement layers.
l	Index for refinement layers, $l \in \{1, \dots, L\}$.
$\mathbf{S}_t \in \mathbb{R}^{d \times d}$	Recurrent memory state at time step t.
$\mathbf{A}_t \in \mathbb{R}^{d \times d}$	Forgetting operator (Linear Decay).
$\mathbf{B}_t \in \mathbb{R}^{d \times d}$	Injection operator (High-Rank Update).
$\mathbf{k}_t^{(l)} \in \mathbb{R}^d$	Geometric projection basis for layer l at time t.
$\mathbf{r}_t^{(l)} \in \mathbb{R}^d$	Input-anchored residual estimate for layer l at time t.
$p_t^{(l)} \in \mathbb{R}^d$	Predicted contextual gain vector for layer l.
$\beta_t^{(l)} \in \mathbb{R}$	Scalar gate (step size) for layer l.

B. Analysis of Degeneracy Under Linear Mapping

This section formally analyzes the representational degeneracy inherent to linear attention models that employ linear projections for both key and value embeddings. We show that under this setup, the optimal state matrix admits a closed-form, sequence-independent solution, rendering online recurrent updates functionally redundant.

Consider a standard linear attention framework, where the key and value vectors for token t are computed as linear transformations of the input token representation $\mathbf{x}_t$:

$$\mathbf{k}_t = \mathbf{W}_k \mathbf{x}_t, \quad \mathbf{v}_t = \mathbf{W}_v \mathbf{x}_t, \tag{19}$$

where $\mathbf{W}_k \in \mathbb{R}^{d \times d_e}$ and $\mathbf{W}_v \in \mathbb{R}^{d \times d_e}$ are learnable projection matrices, and d_e denote the dimensions of the input vector. The model maintains a recurrent state matrix $\mathbf{S}_t$, updated online to minimize the per-step reconstruction objective:

$$\mathcal{L}_t(\mathbf{S}_{t-1}) = \frac{1}{2} \left\| \mathbf{S}_{t-1} \mathbf{k}_t - \mathbf{v}_t \right\|_2^2 . \tag{20}$$

Assume that the key projection matrix $\mathbf{W}_k$ is square and invertible, i.e., $d = d_e$ and $\det(\mathbf{W}_k) \neq 0$. For each token t, the objective $\mathcal{L}_t(\mathbf{S}_{t-1})$ is a convex quadratic function in $\mathbf{S}_t$, and its unique global minimum can be obtained in closed form by setting the gradient with respect to $\mathbf{S}_t$ to zero. The gradient is derived as:

$$\nabla_{\mathbf{S}_t} \mathcal{L}_t(\mathbf{S}_{t-1}) = (\mathbf{S}_{t-1} \mathbf{k}_t - \mathbf{v}_t) \mathbf{k}_t^\top . \tag{21}$$

Setting the gradient to zero, we have:

$$\mathbf{S}_t \mathbf{k}_t \mathbf{k}_t^\top = \mathbf{v}_t \mathbf{k}_t^\top . \tag{22}$$

Substituting the linear definitions of $\mathbf{k}_t$ and $\mathbf{v}_t$ and leveraging the invertibility of $\mathbf{W}_k$, we obtain the optimal state matrix for all t:

$$\mathbf{S}^\star = \mathbf{W}_v \mathbf{W}_k^{-1} . \tag{23}$$

This solution exhibits two critical properties. It is time-invariant, as $\mathbf{S}^\star$ bears no dependence on the time index t, ensuring the optimal state remains static throughout the entire sequence. It is also sequence-independent, determined exclusively by the projection parameters $\mathbf{W}_k$ and $\mathbf{W}_v$ with no relation to the content, order, or length of the input sequence.

This outcome reveals a fundamental degeneracy within the framework. The recurrent state matrix fails to encode sequence-specific or temporal information, functioning merely to realize a fixed linear mapping between the key and value spaces. Even though online update mechanisms such as gradient descent and exponential moving averages can be numerically implemented to approximate this solution, they add no meaningful representational capacity for modeling sequential patterns.

One way to overcome this degeneracy is to introduce a nonlinear transformation into the mapping from the state-driven prediction $\mathbf{S}_{t-1} \mathbf{k}_t$ to the target $\mathbf{v}_t$.

Algorithm 1 PRISM Forward Pass: Input-Anchored Parallel Prediction

Input: Sequence $\mathbf{X} \in \mathbb{R}^{B \times N \times D}$, Parameters Θ
// PHASE 1: INPUT-ANCHORED SIMULATION (PARALLEL)
Construct proxies for state-dependent terms using local context.
$\mathbf{U} \leftarrow \text{SiLU}(\text{ShortConv}(\mathbf{X}))$ {The Anchor: Proxy for $\mathbf{S}_{t-1}\mathbf{k}_t$}
$\mathbf{Q}, \mathbf{V} \leftarrow \text{Linear}(\mathbf{U})$
$\boldsymbol{\alpha} \leftarrow \sigma(\text{Linear}_\alpha(\mathbf{U}))$ {Linear Decay Rate}
Compute Ideal Solver Components for $l = 1 \dots L$:
for $l = 1$ **to** L **in parallel do**
  $\mathbf{K}^{(l)} \leftarrow \text{Linear}_k^{(l)}(\mathbf{U})$ {Projection Basis}
  $\mathcal{P}^{(l)} \leftarrow \text{Linear}_P^{(l)}(\mathbf{U})$ {Gain Simulator ($\approx \sigma'$)}
  $\beta^{(l)} \leftarrow \text{Linear}_\beta^{(l)}(\mathbf{U})$ {Scalar Step Size}
end for
Initialize Residual Proxy:
$\mathbf{R}^{(1)} \leftarrow \mathbf{V} - \mathbf{U}$
// PHASE 2: FUSED RANK ACCUMULATION (SRAM RESIDENT)
Inner loop fused into the recurrent kernel.
for $t = 1$ **to** N **do**
  // 1. Load Step Parameters
  $\mathbf{q}_t, \mathbf{v}_t, \alpha_t \leftarrow \mathbf{Q}, \mathbf{V}, \boldsymbol{\alpha}$
  // 2. Iterative Solver Loop
  Register: $\mathbf{r}_t \leftarrow \mathbf{R}^{(1)}; \mathbf{B}_t \leftarrow 0$
  for $l = 1$ **to** L **do**
    *// Simulate Gradient Step: Gain * Residual*
    $\boldsymbol{\delta}_t \leftarrow \text{GELU}(p_t^{(l)} \odot \mathbf{r}_t)$
    // Accumulate High-Rank Injection
    $\mathbf{B}_t \leftarrow \mathbf{B}_t + \beta_t^{(l)} \cdot (\boldsymbol{\delta} \otimes \mathbf{k}_t^{(l)})$
    // Orthogonalize Residual
    $\mathbf{r}_t \leftarrow \mathbf{r}_t - \boldsymbol{\delta}_t$
  end for
  // 3. Recurrent State Update (Linear Scan)
  $\mathbf{k}_{decay} \leftarrow \mathbf{k}_t^{(1)}$ {Reuse Layer 1 Basis}
  $\mathbf{v}_{decay} \leftarrow \beta_t^{(1)} \mathbf{S}_{t-1} \cdot \mathbf{k}_{decay}$
  $S_t \leftarrow \alpha(S_{t-1} - \mathbf{v}_{decay} \otimes \mathbf{k}_{decay}) + \mathbf{B}_t$
  $\mathbf{y}_t \leftarrow \text{OutProj}(\mathbf{S}_t \cdot \mathbf{q}_t)$
end for

C. Ideal Solver Derivation and Parallelism Constraints

In this section, we derive the ideal update rule for sequence modeling and show that its explicit dependence on the hidden state $\mathbf{S}_{t-1}$ violates the state-independence requirement for parallel prefix scanning. We then demonstrate that reformulating the recurrence in terms of state-independent operators $\mathbf{A}_t$ and $\mathbf{B}_t$, i.e., ensuring $\mathbf{A}_t, \mathbf{B}_t$ depend only on the input—restores associativity and enables efficient parallelization.

C.1. Derivation of the Ideal Non-Linear Delta Rule

Let the memory state $\mathbf{S} \in \mathbb{R}^{d \times d}$ parameterize a non-linear layer $\hat{\mathbf{v}}_t = \sigma(\mathbf{S}_{t-1}\mathbf{k}_t)$, where σ is a non-linear activation. Given a target value $\mathbf{v}_t$, the instantaneous reconstruction objective is:

$$\mathcal{L}_t(\mathbf{S}_{t-1}) = \frac{1}{2}\|\sigma(\mathbf{S}_{t-1}\mathbf{k}_t) - \mathbf{v}_t\|_2^2, \tag{24}$$

We seek an update direction that reduces the instantaneous loss. Applying a first-order gradient descent step with respect to the hidden state yields

$$\begin{aligned}\Delta \mathbf{S}_t &= -\beta \frac{\partial \mathcal{L}_t(\mathbf{S}_{t-1})}{\partial \mathbf{S}} \\ &= \beta \cdot \underbrace{(\mathbf{v}_t - \sigma(\mathbf{S}_{t-1}\mathbf{k}_t))}_{\text{Residual}} \odot \underbrace{\sigma'(\mathbf{S}_{t-1}\mathbf{k}_t)}_{\text{Contextual Gain}} \cdot \underbrace{\mathbf{k}_t^\top}_{\text{Basis}}\end{aligned} \tag{25}$$

where β is the step size. This expression reveals the canonical structure of an expressive recurrent update, combining prediction error as residual, context-dependent modulation as gain, and projection direction as basis.

C.2. Analysis of Parallel

For a linear recurrence $\mathbf{S}_t = \mathbf{S}_{t-1}\mathbf{A}_t + \mathbf{B}_t$, we define the transition tuple $\theta_t = (\mathbf{A}_t, \mathbf{B}_t)$. The composition of two steps t and $t+1$ is well-defined:

$$\theta_{t:t+1} = \theta_{t+1} \odot \theta_t = (\mathbf{A}_t\mathbf{A}_{t+1}, \mathbf{B}_t\mathbf{A}_{t+1} + \mathbf{B}_{t+1}) \tag{26}$$

Following Definition 3.1, if this update rule can be computed in parallel, then the composed operator $\theta_{t:t+1}$ must depend only on the inputs $\mathbf{x}_{\leq t+1}$. For ideal solver, we have

$$\mathbf{A}_t = \mathbf{I}, \quad \mathbf{B}_t = (\mathbf{v}_t - \sigma(\mathbf{S}_{t-1}\mathbf{k}_t)) \odot \sigma'(\mathbf{S}_{t-1}\mathbf{k}_t) \cdot \mathbf{k}_t^\top \tag{27}$$

Note that both the nonlinear activation σ and its derivative σ' are evaluated at $\mathbf{S}_{t-1}\mathbf{k}_t$. Consequently, the correction term $\mathbf{B}_t$ is an explicit function of the hidden state $\mathbf{S}_{t-1}$, making the recurrence state-dependent. This violates the requirement in Definition 3.1, which mandates that the transition operators $\mathbf{A}_t$ and $\mathbf{B}_t$ depend solely on the input $\mathbf{x}_{\leq t}$ and not on any intermediate state.

Standard linear attention approximates the gain term in (25) as $\mathbf{1}$ and omit the non-linearity $\sigma(\mathbf{S}_{t-1}\mathbf{k}_t)$ in residual. While this simplification ensures state-independent operators and thus satisfies the parallelism condition in Definition 3.1, it comes at the cost of reduced expressivity—effectively discarding the very mechanisms that enable dynamic, input-aware memory updates. Crucially, however, Definition 3.1 is satisfied as long as $\mathbf{A}_t$ and $\mathbf{B}_t$ are *state-independent*, regardless of how expressive they are. This observation forms the foundation of PRISM: rather than computing the ideal update directly, we design a state-independent predictor for $\mathbf{B}_t$ that approximates the residual and gain terms of the ideal solver using only local input context. By anchoring the simulation to the current token and decoupling write and forget dynamics, PRISM recovers the essential structure of the ideal rule while strictly adhering to the parallelism constraint.

D. Sensitivity Analysis of Write-Forget Dynamics via Spectral Perturbation

In this section, we rigorously analyze the error propagation properties of the Write-Forget Decoupling strategy. We establish the theoretical foundation by bridging HiPPO theory with discrete sequence modeling via Zero-Order Hold (ZOH) discretization, and then perform a worst-case spectral perturbation analysis to bound the total accumulated error.

D.1. Theoretical Foundation: HiPPO, ODEs, and ZOH

We posit that the optimal strategy for sequence modeling is to approximate a continuous-time measure update process.

The HiPPO Formalism. As established in HiPPO (Gu et al., 2020), the problem of online function approximation over a history window can be cast as an Ordinary Differential Equation (ODE):

$$\dot{\mathbf{h}}(t) = \mathcal{A}(t)\mathbf{h}(t) + \mathcal{B}(t)\mathbf{x}(t) \tag{28}$$

where the state $\mathbf{h}(t)$ represents the coefficients of the input history projected onto a basis of orthogonal polynomials (e.g., Legendre or Laguerre). The operator $\mathcal{A}(t)$ governs the "forgetting" or compression of history.

Justification for Zero-Order Hold (ZOH). In language modeling, the input $\mathbf{x}(t)$ is not a continuous signal but a sequence of discrete tokens arriving at times t_k. The standard method to discretize Eq. (28) while preserving the underlying continuous memory characteristics is the Zero-Order Hold (ZOH) assumption, which posits that the input $\mathbf{x}(t)$ is constant during the interval $[t_{k-1}, t_k)$. This yields the discrete recurrence:

$$\mathbf{h}_k = \mathbf{A}_k\mathbf{h}_{k-1} + \mathbf{B}_k\mathbf{x}_k \tag{29}$$

where $\mathbf{A}_k = \exp(\Delta\mathcal{A})$ and $\mathbf{B}_k = (\mathbf{A}_k - \mathbf{I})\mathcal{A}^{-1}\mathcal{B}$. Consequently, the discrete matrix $\mathbf{A}_k$ inherits the spectral stability properties of the continuous operator $\mathcal{A}$. PRISM approximates this discrete $\mathbf{A}_k$ directly.

D.2. Spectral Bounds via Gated DeltaNet

A critical requirement for stability is that the eigenvalues of $\mathbf{A}_k$ must lie within the unit circle. PRISM adopts the decay parameterization from Gated DeltaNet.

Lemma D.1 (Bounded Spectrum of Gated DeltaNet)**.** *The forgetting operator in PRISM is parameterized as* $\mathbf{A}_t = \mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top$*, where* $\beta_t \in [0, 1]$ *and* $\|\mathbf{k}_t\| \leq 1$*. The eigenvalues of this matrix satisfy:*

$$\lambda(\mathbf{A}_t) \in [-1, 1] \tag{30}$$

Proof. We analyze the spectrum of $\mathbf{A}_t$ by considering the eigenspace aligned with $\mathbf{k}_t$ and its orthogonal complement:

1. **Orthogonal Subspace:** For any vector $\mathbf{v}$ such that $\mathbf{v} \perp \mathbf{k}_t$ (spanning $d - 1$ dimensions), we have:

$$\mathbf{A}_t \mathbf{v} = \mathbf{v} - \beta_t \mathbf{k}_t (\mathbf{k}_t^\top \mathbf{v}) = \mathbf{v} \tag{31}$$

   Thus, there are $d - 1$ eigenvalues equal to 1.

2. **Principal Subspace:** For the vector $\mathbf{v} = \mathbf{k}_t$, we have:

$$\mathbf{A}_t \mathbf{k}_t = \mathbf{k}_t - \beta_t \mathbf{k}_t (\mathbf{k}_t^\top \mathbf{k}_t) = (1 - \beta_t \|\mathbf{k}_t\|^2)\mathbf{k}_t \tag{32}$$

   The remaining eigenvalue is $\lambda_{min} = 1 - \beta_t \|\mathbf{k}_t\|^2$.

Given the constraints $0 \leq \beta_t \leq 1$ and $0 \leq \|\mathbf{k}_t\|^2 \leq 1$, the term $\beta_t \|\mathbf{k}_t\|^2$ lies in $[0, 1]$. Consequently, $\lambda_{min} \in [0, 1] \subset [-1, 1]$. This structural guarantee ($\rho(\mathbf{A}_t) \leq 1$) ensures the system is non-expansive and prevents exponential divergence. □

D.3. Decomposition of Cumulative Error Terms

We now analyze the impact of using a Low-Rank approximation for $\mathbf{A}$. Let the ideal operator be $\mathbf{A}^*$ and the PRISM approximation be $\hat{\mathbf{A}} = \mathbf{A}^*(I + \Delta)$. The total state error $\mathbf{e}_T$ at step T is the sum of error contributions from all previous time steps k. We decompose this into Transmission Error (from $\Delta\mathbf{A}$) and Injection Error (from $\Delta\mathbf{B}$).

$$\mathbf{e}_T = \sum_{k=1}^{T} \mathbf{e}_{path}(T, k) + \sum_{k=1}^{T} \mathbf{e}_{inj}(T, k) \tag{33}$$

For the Transmission Error, we focus on a single invariant subspace with eigenvalue λ. The variance of the error contribution from lag $\tau = T - k$ is derived using the unbiased log-normal perturbation model:

$$V(\tau, \lambda) \coloneqq \text{Var}(\mathcal{E}_{path}(\tau)) \approx \tau |\lambda|^{2\tau} \sigma^2 f(\lambda) \tag{34}$$

where $\sigma^2 f(\lambda)$ is the variance of the relative perturbation in the eigenvalue.

D.4. Robust Error Bound via Max-Sum Analysis

We seek a robust upper bound for the total error. **Assumption (Quadratic Scaling of Floating Point Error).** We assume the numerical precision of the decay rate is bounded by relative error. Since $\lambda = 1 - \beta$, a relative error in β implies the noise variance scales quadratically with the decay magnitude $\gamma = 1 - |\lambda|$ (or $\gamma \approx -\ln|\lambda|$):

$$\sigma^2 f(\lambda) \leq K\gamma^2 \tag{35}$$

Substituting this into the variance equation:

$$V(\tau, \gamma) \leq \tau e^{-\gamma\tau}(K\gamma^2) \tag{36}$$

Theorem D.2 (Logarithmic Worst-Case Stability)**.** *The worst-case accumulated error of the Forgetting operator at step T, maximizing over possible decay rates, grows at most logarithmically.*

Proof. The proof proceeds in two steps:

1. Define Worst-Case Envelope $g(\tau)$. We aim to find the decay rate γ^* that maximizes the error at a specific lag τ. Let $h(\gamma) = K\tau\gamma^2 e^{-\gamma\tau}$. Differentiating with respect to γ and setting the derivative to zero yields the critical point $\gamma^* = 2/\tau$. Substituting γ^* back into the variance function V:

$$g(\tau) = \max_{\gamma} V(\tau, \gamma) = K\tau \left(\frac{2}{\tau}\right)^2 e^{-2} = \frac{4Ke^{-2}}{\tau} \tag{37}$$

2. Summation Bound. The total error variance at step T is bounded by the sum of these worst-case envelopes:

$$\sum_{\tau=1}^{T} g(\tau) \propto \sum_{\tau=1}^{T} \frac{1}{\tau} \approx \ln T \tag{38}$$

Thus, we conclude that $\text{Var}(\mathbf{e}_T) \leq O(\ln T)$. □

While the worst-case error grows logarithmically, the **average** behavior is even more stable. We now show that the total error energy (sum of variances over time) is linear in T, implying the average error per step is constant.

Theorem D.3 (Constant Average-Case Error)**.** *For any fixed mode λ (and thus fixed decay rate γ), the total accumulated error energy over T steps is $O(T)$, implying that the average error variance per step is $O(1)$.*

Proof. Let $\mathcal{V}_{total} = \sum_{t=1}^{T} \text{Var}(\mathbf{e}_t)$ denote the total error energy accumulated over the entire sequence. We exchange the order of summation to count the total contribution of each individual injection $\mathbf{B}_k$ throughout its lifetime:

$$\mathcal{V}_{total} = \sum_{t=1}^{T} \sum_{k=1}^{t} V(t-k, \gamma) = \sum_{k=1}^{T} \underbrace{\sum_{\tau=0}^{T-k} V(\tau, \gamma)}_{\text{Lifetime Error of } \mathbf{B}_k} \tag{39}$$

Consider the inner sum for a fixed γ. Recall that $V(\tau, \gamma) \leq \tau e^{-\gamma\tau}(K\gamma^2)$. We upper bound the finite sum by the infinite series. Using the identity $\sum_{\tau=0}^{\infty} \tau r^\tau = \frac{r}{(1-r)^2}$ with $r = e^{-\gamma}$:

$$\sum_{\tau=0}^{\infty} V(\tau, \gamma) \leq K\gamma^2 \sum_{\tau=0}^{\infty} \tau (e^{-\gamma})^\tau = K\gamma^2 \frac{e^{-\gamma}}{(1-e^{-\gamma})^2} \tag{40}$$

For small γ (long context), we use the Taylor approximation $1 - e^{-\gamma} \approx \gamma$ and $e^{-\gamma} \approx 1$:

$$\sum_{\tau=0}^{\infty} V(\tau, \gamma) \approx K\gamma^2 \cdot \frac{1}{\gamma^2} = K \tag{41}$$

This cancellation is crucial. It demonstrates that the total error contributed by any single injection $\mathbf{B}_k$ over its entire lifetime is bounded by a constant K, independent of the decay rate γ. Finally, summing over all T injections:

$$\mathcal{V}_{total} \leq \sum_{k=1}^{T} K = O(T) \implies \text{Average}(\text{Var}(\mathbf{e}_t)) = \frac{\mathcal{V}_{total}}{T} = O(1) \tag{42}$$

This concludes that while the worst-case error at a specific adversarial step may scale with $\ln T$, the average error remains constant for any stable dynamics. □

D.5. The Necessity of Rank-L Injection (B-Perturbation)

In contrast, consider the Injection Error (Term B). The error variance at lag τ is simply $e^{-\gamma\tau}\sigma_B^2$. The worst case is trivial to achieve: simply set $\gamma \to 0$ (i.e., $\lambda \to 1$), which corresponds to persistent memory.

$$g_B(\tau) = \max_{\gamma} e^{-\gamma\tau}\sigma_B^2 = \sigma_B^2 \tag{43}$$

Summing this over the sequence yields:

$$\mathrm{Var}(\mathbf{e}_T^{(B)}) = \sum_{\tau=1}^{T} \sigma_B^2 = T \cdot \sigma_B^2 \tag{44}$$

Final Comparison:

- **A-Error (Forgetting):** Worst-case $O(\ln T)$, Average-case $O(1)$.
- **B-Error (Writing):** Worst-case $O(T)$ (Linear, simply set $\lambda \to 1$).

This provides a decisive theoretical justification for PRISM: The system is exponentially more robust to approximations in A than in B. Therefore, computational budget (Rank) must be prioritized for B (via Iterative Residual Fitting) rather than A.

E. Simulating State-Dependence via Structural Alignment

The ideal update is driven by two terms derived from the pre-activation vector $\mathbf{z}_t = \mathbf{S}_{t-1}\mathbf{k}_t$. We now analyze the error incurred by replacing the unavailable global interaction $\mathbf{z}_t$ with a local proxy $\mathbf{u}_t = \mathrm{ShortConv}(\mathbf{x}_{\le t})$ in PRISM.

Assume the linear recurrence follows a stable decay pattern (Fading Memory). The true pre-activation vector can be expanded as:

$$\mathbf{z}_t = \mathbf{S}_{t-1}\mathbf{k}_t = \left(\sum_{i=0}^{t-1} \gamma^{t-i}\mathbf{v}_i\mathbf{k}_i^\top\right)\mathbf{k}_t, \tag{45}$$

where $\gamma \in (0,1)$ is the effective decay rate. Assume the ShortConv proxy effectively captures the most recent terms through a local window of size w:

$$\mathbf{u}_t = \left(\sum_{i=t-w}^{t-1} \gamma^{t-i}\mathbf{v}_i\mathbf{k}_i^\top\right)\mathbf{k}_t \tag{46}$$

and $\|\mathbf{v}_i\mathbf{k}_i^\top\mathbf{k}_t\|_2 \le 1$. Then, the approximation error is dominated by the "tail" of the sequence:

$$\|\mathbf{z}_t - \mathbf{u}_t\|_2 \le \left\|\sum_{i=0}^{t-w-1} \gamma^{t-i}\mathbf{v}_i\mathbf{k}_i^\top\mathbf{k}_t\right\|_2 \le \sum_{i=0}^{t-w-1} \gamma^{t-i} \le \frac{\gamma^{w+1}}{1-\gamma}. \tag{47}$$

Thus, the error between the global state projection and the local proxy decays exponentially with the kernel size w.

F. Rank Expansion via Multi-Component Injection

We analyze the rank of the total injection matrix $\mathbf{B}_t$ in PRISM. The update at time t is constructed as a sum of L rank-1 components:

$$\mathbf{B}_t = \sum_{l=1}^{L} \beta_t^{(l)} \cdot \boldsymbol{\delta}_t^{(l)}\mathbf{k}_t^{(l)\top}, \tag{48}$$

where each term is an outer product scaled by a scalar gain $\beta_t^{(l)}$. By the sub-additivity of matrix rank,

$$\mathrm{rank}(\mathbf{B}_t) \le \sum_{l=1}^{L} \mathrm{rank}\big(\boldsymbol{\delta}_t^{(l)}\mathbf{k}_t^{(l)\top}\big) = L. \tag{49}$$

This bound is tight when the vectors $\{\boldsymbol{\delta}_t^{(l)}\}_{l=1}^{L}$ (or $\{\mathbf{k}_t^{(l)}\}_{l=1}^{L}$) are linearly independent.

In contrast, standard linear attention uses a single outer product $\mathbf{v}_t\mathbf{k}_t^\top$, yielding $\mathrm{rank}(\mathbf{B}_t) = 1$. PRISM expands the reachable update directions to a structured low-dimensional cone spanned by $\{\mathbf{k}_t^{(1)}, \ldots, \mathbf{k}_t^{(L)}\}$. This allows the model to perform multi-modal updates—simultaneously modifying orthogonal semantic subspaces (e.g., updating 'Gender' and 'Tense' attributes independently) within a single timestep.

G. Stability of Nested Non-Linear Refinement

Stacking multiple non-linear layers ($\boldsymbol{\delta} = \mathrm{GELU}(p \odot r)$) inside a recurrent loop may leads to gradient explosion or instability. We provide a stability bound via perturbation analysis.

Lemma F.1 (Stability under Perturbation). Consider a single refinement step. Let $r^{(l)}$ be the nominal residual vector, and $\tilde{r}^{(l)} = r^{(l)} + \epsilon$ be a perturbed residual vector (e.g., due to numerical precision or upstream approximation errors). The update function is $\boldsymbol{\delta}^{(l)} = \mathrm{GELU}(\boldsymbol{p}^{(l)} \odot \boldsymbol{r}^{(l)})$. Assume the activation function (GELU) is L_ϕ-Lipschitz continuous (where $L_\phi \approx 1$) and the predictor outputs are bounded by $\|\boldsymbol{p}^{(l)}\|_\infty \leq M$ (ensured by normalization).

We analyze the divergence of the update vectors δ and $\tilde{\delta}$:

$$\|\boldsymbol{\delta}^{(l)} - \tilde{\boldsymbol{\delta}}^{(l)}\| = \|\mathrm{GELU}(p^{(l)} \odot \mathbf{r}^{(l)}) - \mathrm{GELU}(p^{(l)} \odot \tilde{\mathbf{r}}^{(l)})\| \tag{50}$$

$$\leq L_\phi \|p^{(l)} \odot (\mathbf{r}^{(l)} - \tilde{\mathbf{r}}^{(l)})\| \tag{51}$$

$$\leq L_\phi M \|\epsilon\| \tag{52}$$

Conclusion: The output perturbation is linearly bounded by the input perturbation scaled by M. Since $M \approx 1$ in practice (due to LayerNorm) and the loop depth L is small (typically $2 \sim 4$), the iterative refinement remains numerically stable. It does not suffer from the chaotic divergence typical of deep unrolled optimizers. The greedy subtraction ($r \leftarrow r - \delta$) further acts as a negative feedback mechanism, naturally bounding the residual energy.

H. Mechanistic Probing Details

To ensure reproducibility and facilitate future research into the "Storage vs. Computation" trade-off, we provide the detailed generation logic for the full suite of mechanistic probing tasks. These tasks are categorized into three groups: Memory Capacity, Non-Linear Logic, and Gating Control.

H.1. Data Generation Logic

We employ a strict **Vocabulary Separation** strategy. The vocabulary V is partitioned into disjoint sets: $\mathcal{V}_{data}$ for operands/values, $\mathcal{V}_{control}$ for triggers/queries, and $\mathcal{V}_{noise}$ for background noise. This prevents the model from relying on simple token-ID memorization.

H.2. Task Instantiation Examples

Table 6 provides concrete input-output examples for all 9 tasks, demonstrating the diverse requirements from simple retrieval to complex arithmetic.

Implementation Note. For Logical tasks (Type II), we ensure that the relevant operands fall within the local receptive field of the ShortConv anchor (or rely on the PRISM gate for state reset in N-bit Parity). This enforces a test of the *update rule's expressivity* (can it approximate the non-linear function?) rather than long-range retrieval capacity.

I. Recommendation Dataset Descriptions

Datasets. We evaluate our model on four widely-used recommendation benchmarks: *Amazon Books*, *Amazon Movies*, *Amazon Electronics*, and *Yelp*. Recommendation is selected as a critical stress test for **High-Rank Sparsity**. Unlike natural language, where attention patterns are often low-rank and semantic-dependent, user interest evolution is inherently multi-modal and diverse. This high-entropy distribution contrasts sharply with the low-rank assumptions of standard linear attention, requiring the model to capture complex, non-linear transition dynamics over long horizons.

Algorithm 2 Comprehensive Synthetic Task Generation

Input: Seq Length N, Vocab V, Window W
Init: Fill sequence X with noise $\sim \mathcal{V}_{noise}$
// TYPE I: ASSOCIATIVE MEMORY (Storage)
Task 1: MQAR (Multi-Query Associative Recall)
Sample K pairs (k_i, v_i). Place them at random positions.
Query: Append k_i. Target: v_i.
Task 2: Poly-Recall (Contextual Disambiguation)
Define contexts C_1, C_2. Map key k to v_1 if C_1, v_2 if C_2.
Place pairs $[C, k, v]$. Query: $[C_{target}, k]$. Target: v_{target}.
Task 3: Variable Tracking
Define chain: $a = val, b = a, c = b$.
Place assignments randomly. Query: c. Target: val.
// TYPE II: NON-LINEAR LOGIC (Reasoning)
Task 4: Local XOR
Sample a, b. Label = 1 if $(a \pmod 2) \neq (b \pmod 2)$ else 0.
Place $[a, b, \text{TOK\_XOR}]$. Query result.
Task 5: N-bit Parity
Sample n bits. Label = $(\sum b_i) \pmod 2$.
Place contiguous block $[b_1, \ldots, b_n]$. Query result.
Task 6: Modulo Addition
Sample a, b. Label = $(a + b) \pmod M$.
Place $[a, b]$. Query result.
Task 7: Palindrome Detection
Sample a, b, c. Label = 1 if $a == c$ else 0.
Place $[a, b, c]$. Query result. (Tests structural sensitivity).
// TYPE III: GATING & CONTROL (Robustness)
Task 8: Silence Gate (Noise Filtering)
Sample trigger $T \in \{ON, OFF\}$.
If $T = ON$: Target $= v$. If $T = OFF$: Target = NULL.
Place $[T, k, v]$. Query k.
Task 9: MUX Logic (Multiplexer)
Sample selector $S \in \{0, 1\}$. Sample channels ch_0, ch_1.
Label $= ch_0$ if $S = 0$ else ch_1.
Place $[S, ch_0, ch_1]$. Query result.

Data Sources and Preprocessing. The Amazon datasets are derived from the Amazon Review Data (2014) collection, encompassing user reviews and metadata spanning from 1996 to 2014. The Yelp dataset is sourced from the Yelp Open Dataset challenge, capturing user interactions with businesses. To rigorously evaluate the capacity of PRISM to handle long-term dependencies, we employ a dense filtering strategy. We retain only items that appear at least **5 times** and users with at least **40 interactions**. As shown in Table 7, this filtering results in sequences with high average lengths (e.g., 121.22 for Amazon Movies), posing a significant challenge for memory retention in linear RNNs.

Evaluation Protocol. We strictly follow the chronological order of user interactions. We adopt the standard **leave-one-out** evaluation strategy: for each user sequence, the most recent interaction is reserved for the *test set*, the second-to-last interaction forms the *validation set*, and all preceding interactions constitute the *training set*.

J. Baseline Model Descriptions

To provide a unified perspective on the evaluated architectures, we analyze all linear baseline models through the lens of the generalized linear recurrence:

$$\mathbf{S}_t = \mathbf{A}_t \mathbf{S}_{t-1} + \mathbf{B}_t \tag{53}$$

Table 6. Examples of Mechanistic Probing Tasks ($D = 16, V = 64$). $\mathcal{N}$ denotes noise tokens. **Logic** tasks highlight the capability gap between Linear Attention and PRISM.

Type	Task	**Input Pattern** (Conceptual)	**Underlying Logic**	**Target**
Memory (Capacity)	MQAR	`k=5,v=9 ... k=5`	Retrieval: $Mem[5] \to 9$	9
	Poly-Recall	`CtxA,k=5,v=9 ... CtxA,k=5`	Contextual: $Mem[A][5] \to 9$	9
	Var. Tracking	`a=7 ... b=a ... c=b ... c=?`	Pointer Chain: $c \to b \to a \to 7$	7
Logic (Reasoning)	Local XOR	`[5, 8, XOR]`	$odd(5) \neq even(8) \to$ True	1
	N-bit Parity	`[1, 1, 1]` (N=3)	$1+1+1=3 \to odd$	1
	Modulo Add	`[8, 4]` ($M = 10$)	$(8+4) \pmod{10} = 12 \to 2$	2
	Palindrome	`[4, 9, 4]`	$First(4) == Last(4) \to$ True	1
Control (Gating)	Silence Gate	`[OFF, k=5, v=9] ... k=5`	Gating: $Gate(OFF) \approx 0 \to$ Ignore	NULL
	MUX Logic	`[Sel=1, Ch0=3, Ch1=8]`	Routing: $Sel = 1 \to$ Pick $Ch1$	8

Table 7. Statistics of the datasets used in our experiments. We adopt a dense filtering setting (User interactions ≥ 40) to stress-test the models on long-range dependency modeling.

Dataset	# Users	# Items	# Interactions	Avg. Length
Amazon_books	31,103	339,960	3,319,359	106.72
Amazon_movies	9,429	58,636	1,142,976	121.22
Amazon_elecs	1,869	33,135	123,147	65.89
Yelp	17,233	126,829	1,605,608	93.17

where $\mathbf{S}_t \in \mathbb{R}^{d \times d}$ is the hidden state (or memory matrix), $\mathbf{A}_t$ governs the decay/transition dynamics, and $\mathbf{B}_t$ represents the information injection at step t. The formulations for each baseline are detailed below.

J.1. Linear Heuristics (Decay-Based)

These models primarily focus on heuristically designing the forgetting mechanism $\mathbf{A}_t$ while using a simple rank-1 injection $\mathbf{B}_t = \mathbf{v}_t \mathbf{k}_t^\top$.

SLA (Simple Linear Attention) (Katharopoulos et al., 2020): Represents the canonical un-gated linear attention. It assumes no forgetting, treating the sequence as an infinite buffer.

- $\mathbf{A}_t = \mathbf{I}$ (Identity)
- $\mathbf{B}_t = \mathbf{v}_t \mathbf{k}_t^\top$

GLA (Gated Linear Attention) (Yang et al., 2024a): Introduces a data-dependent scalar or diagonal decay to handle local shifts in context.

- $\mathbf{A}_t = \text{diag}(\alpha_t)$, where $\alpha_t = \sigma(\mathbf{W}_\alpha \mathbf{x}_t)$
- $\mathbf{B}_t = \beta_t(\mathbf{v}_t \mathbf{k}_t^\top)$, where β_t is an input-dependent gate.

Mamba-2 (State Space Duality) (Dao & Gu, 2024): Optimizes the recurrence for hardware efficiency using the SSD (Structured State Space Duality) framework. It can be viewed as a linear attention with a structured mask.

- $\mathbf{A}_t = \text{diag}(\exp(-\Delta_t \mathbf{A}))$, representing discretized input-dependent decay.
- $\mathbf{B}_t = \mathbf{v}_t \mathbf{k}_t^\top$ (In SSD form).

GSA (Gated Slot Attention) (Zhang et al., 2024): Refines the gating mechanism by treating memory dimensions as independent "slots". It typically enforces a normalization or complementarity between forgetting and writing (e.g., via Softmax or $\mathbf{f}_t \approx 1 - \mathbf{i}_t$) to simulate a smooth cache replacement policy.

- $\mathbf{A}_t = \text{diag}(\mathbf{f}_t)$, where $\mathbf{f}_t$ is the slot retention gate.
- $\mathbf{B}_t = \mathbf{i}_t \odot (\mathbf{v}_t \mathbf{k}_t^\top)$, where $\mathbf{i}_t$ is the slot input gate.

MoM (Mixture of Memories) (Du et al., 2025): Enhances capacity by utilizing multiple independent memory heads, each following a standard GLA update rule.

- $\mathbf{S}_t = \sum_{i=1}^{H} \text{Head}_i(\mathbf{S}_{t-1}^{(i)})$, where each head evolves with $\mathbf{A}_t^{(i)} = \text{diag}(\alpha_t^{(i)})$.

J.2. Linear Optimization (Delta Rule)

These models perform online optimization on a **Linear** objective, allowing the update to be written exactly in the recursive form $\mathbf{S}_t = \mathbf{A}_t\mathbf{S}_{t-1} + \mathbf{B}_t$.

Gated DeltaNet (Yang et al., 2024a): Minimizes a linear regression loss $\|\mathbf{v}_t - \mathbf{S}_{t-1}\mathbf{k}_t\|^2$.

- $\mathbf{A}_t = \mathbf{I} - \beta_t\mathbf{k}_t\mathbf{k}_t^\top$ (Update Direction)
- $\mathbf{B}_t = \beta_t\mathbf{v}_t\mathbf{k}_t^\top$ (Injection)

J.3. Non-Linear Optimization (Neural Memory)

These models employ a deep neural network as the memory or predictor. The update rule involves backpropagating through non-linearities (e.g., MLPs), meaning the gradient $\nabla\mathcal{L}$ is a **non-linear function** of the state $\mathbf{S}_{t-1}$. Thus, they **cannot** be formulated as a standard linear recurrence ($S_t = A_tS_{t-1} + B_t$).

TTT-NN (Test-Time Training with Neural Net) (Sun et al., 2024): The hidden state $\mathbf{S}_t$ represents the weights of a two-layer MLP. The update is a step of GD on a non-convex loss:

$$\mathbf{S}_t = \mathbf{S}_{t-1} - \eta \cdot \nabla_{\mathbf{S}}\ell(\text{MLP}(\mathbf{x}_t; \mathbf{S}_{t-1}), \mathbf{y}_t) \tag{54}$$

The term $\nabla_{\mathbf{S}}\ell$ introduces explicit non-linearity, enabling higher expressivity but preventing global parallel prefix scans (Parallelism is restricted to batch/chunk level).

TITANS (Neural Memory) (Behrouz et al., 2024): The memory is parameterized as a deep neural network $\mathcal{M}_{\theta_t}$. The update is governed by a "surprise" metric (gradient magnitude) and includes an adaptive forgetting mechanism:

$$\mathbf{S}_t = (1 - \alpha_t(\mathbf{x}_t))\mathbf{S}_{t-1} - \eta_t(\mathbf{x}_t) \cdot \nabla_{\mathbf{S}}\|\mathcal{M}(\mathbf{k}_t; \mathbf{S}_{t-1}) - \mathbf{v}_t\|^2 \tag{55}$$

Unlike linear models, the gradient term depends non-linearly on $\mathbf{S}_{t-1}$, allowing the model to perform complex associative recall and filtering that linear projections cannot capture.

J.4. Contrast with PRISM

PRISM occupies a unique middle ground. It adopts the **high-rank** update philosophy of optimization models (like TTT/TITANS) but enforces a **linear structural constraint** via the Input-Anchored Proxy.

- Unlike TTT-NN/TITANS, PRISM's update operators $(\mathbf{A}_t, \mathbf{B}_t)$ are computed **independently** of $\mathbf{S}_{t-1}$ (using local convolution).
- This approximation allows PRISM to enjoy the **fully parallel training** of linear models (via $\mathbf{S}_t = \mathbf{A}_t\mathbf{S}_{t-1} + \mathbf{B}_t$) while approximating the **multi-step optimization trajectory** of non-linear solvers.

K. Discussion

In this work, we introduced PRISM to resolve the linearity bottleneck in efficient sequence modeling. Here, we analyze the structural trade-offs and future implications of our approach.

Rank-L Density vs. Rank-1 Sparsity. While standard linear models (e.g., DeltaNet, Mamba) achieve $O(N)$ efficiency, they are structurally confined to Rank-1 updates. This creates a "sparsity bottleneck" where the model can only encode a single feature relation per time step. PRISM overcomes this not by abandoning linearity, but by *densifying* the update operator. By unrolling an input-anchored solver, PRISM enables a Rank-Up-To-L injection within a single step. This

provides a significantly stronger writing capability—allowing the model to capture multi-modal semantic dependencies (e.g., polysemy or compound concepts) without sacrificing the parallel scan property.

The Trade-off: Local Approximation vs. Global Reactivity. By decoupling the solver from the exact recurrent state S_{t-1}, PRISM accepts a fundamental trade-off. The ShortConv proxy effectively captures the high-frequency local gradients required for synthesis, but it is blind to long-term memory artifacts (the "Long Tail"). Consequently, PRISM loses the ability to perform *reactive* error correction based on distant history (e.g., checking if a fact was already stored 4k tokens ago). We explicitly prioritize parallelism over this reactivity, positing that for generative modeling, predicting the optimization trajectory from local context is a highly efficient structural prior.

The Memory Capacity Wall. Our analysis of Write-Forget Decoupling (Appendix B) reveals a deeper theoretical boundary for all linear recurrences. We showed that the forgetting operator $\mathbf{A}_t$ is robust to linearization, implying that complex non-linear gating yields diminishing returns for retention. This suggests that memory overwriting is inevitable in fixed-dimensional states ($S \in \mathbb{R}^{d\times d}$), regardless of the writing algorithm. PRISM maximizes the *quality* of what is written (Fidelity), but it does not expand the *container* (Capacity). This highlights a necessary direction for future research:

- **Capacity Expansion:** To solve the "overwriting" problem, the state capacity must be expanded explicitly, such as via Mixture-of-Memory (MoM)(Du et al., 2025) or GSA(Zhang et al., 2024). PRISM is complementary to these approaches, serving as a high-density writing operator that works synergistically with expanded memory slots.
- **Hybrid Architectures:** The theoretical "blindness" of the input-anchored proxy explains the empirical necessity of Hybrid Architectures (e.g., Jamba)(Lieber et al., 2024). A Transformer layer acts as a periodic non-linear oracle, correcting the long-tail drift that local linear approximations cannot resolve(Team et al., 2025).

Table 8. Recommendation benchmark performance. Metrics are Hit@500 and NDCG@500 with per-dataset AUC. Best linear-attention results are in **bold** and second-best are underlined.

Model	Amazon_books			Amazon_movies			Amazon_elec			Yelp			Mean Rank
	H@500	NDCG@500	AUC	H@500	NDCG@500	AUC	H@500	NDCG@500	AUC	H@500	NDCG@500	AUC	
Linear Attention													
SLA	0.2211	0.0342	0.8866	0.2111	0.0344	0.7461	0.2454	0.0382	0.7023	0.3219	0.0502	0.9392	5.88
GLA	0.1819	0.0270	0.8752	0.2122	0.0333	0.7478	0.2628	0.0381	0.7008	0.2186	0.0346	0.8943	7.62
MoM	0.1700	0.0259	0.8705	0.2322	0.0392	0.7705	0.2410	0.0367	0.7042	0.3106	0.0482	0.9346	8.25
GSA	0.2346	0.0367	0.8870	0.2090	0.0333	0.7427	0.2587	0.0380	0.7087	0.3164	0.0491	0.9383	6.25
MAMBA2	0.2353	0.0368	0.8872	0.2388	0.0398	0.7713	0.2519	0.0378	0.7157	0.3200	0.0495	0.9385	4.25
ATLAS	0.2330	0.0359	0.8884	0.2376	0.0399	0.7710	**0.2629**	0.0388	0.7042	0.3158	0.0487	0.9383	4.25
GDeltanet	0.2275	0.0357	0.8844	0.2212	0.0369	0.7504	0.2424	0.0371	**0.7159**	0.3163	0.0490	0.9367	7.00
TTT	**0.2398**	0.0371	0.8871	0.2254	0.0372	0.7591	0.2403	0.0360	0.6946	0.3140	0.0486	0.9375	6.50
TITANS	0.2362	**0.0374**	0.8869	0.2331	0.0394	0.7652	0.2628	**0.0389**	0.7007	**0.3256**	**0.0505**	**0.9395**	**2.25**
PRISM	0.2383	0.0373	**0.8888**	**0.2407**	**0.0409**	**0.7727**	0.2613	0.0380	0.7134	0.3204	0.0497	0.9393	2.75
Transformer													
SASRec	0.2225	0.0345	0.8910	0.2281	0.0366	0.7677	0.2711	0.0425	0.7293	0.3279	0.0511	0.9410	–
HSTU	0.2310	0.0363	0.8835	0.2385	0.0411	0.7748	0.2574	0.0389	0.7189	0.3093	0.0475	0.9324	–