Improved State Mixing in Higher-order and Block Diagonal Linear Recurrent Networks

Igor Dubinin[1,2], **Antonio Orvieto**[3,4,5], **Felix Effenberger**[1,2]

[1]Natural Intelligence (NISYS GmbH), [2]Ernst Strüngmann Institute for Neuroscience, [3]ELLIS Institute Tübingen, [4]MPI for Intelligent Systems, [5]Tübingen AI Center

Paper: arXiv:2602.12021

Introduction

Linear RNNs and SSMs [1, 2] owe their efficiency to **diagonal** state transitions, which provably limit what they can compute [3, 4]; dense transitions (LSTMs) are expressive but too costly to compute. **Can we buy expressivity without giving up efficiency?** We add **structured state mixing** between the two extremes, across channels (**BD-LRU**) and across time (**H-LRU**), with a **joint gate normalization** that lets both scale with block size m, and **parallel scans** that keep training efficient.

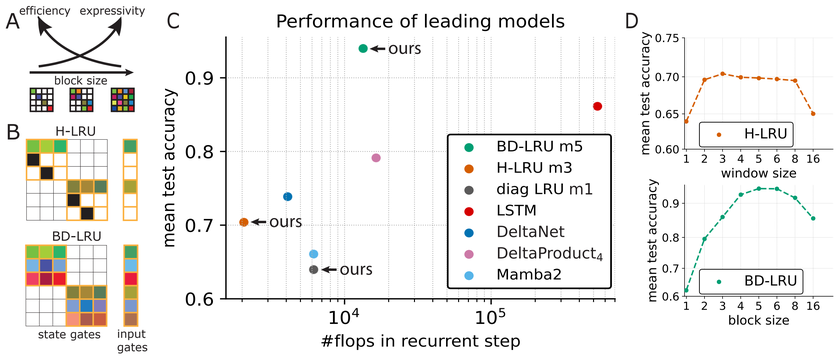


Figure 1: A. The expressivity–efficiency trade-off: block size m moves between diagonal and dense transitions. **B.** Selective gates in block-diagonal form (colored = selective, black = fixed one, white = fixed zero). **C.** Mean accuracy over four synthetic task families vs. FLOPs per step: BD-LRU matches or beats LSTM [5], Mamba2 [6], DeltaNet [7] and DeltaProduct [8] at lower cost. **D.** Performance scales with window/block size m.

Architectures

BD-LRU: dense **channel mixing** inside each block k (N blocks of size m, $H = Nm$):

$$\mathbf{h}_t^k = \mathbf{A}_t^k \mathbf{h}_{t-1}^k + \mathbf{a}_{0,t}^k \odot \mathbf{v}_t^k, \qquad \text{(BD-LRU)}$$

H-LRU: **temporal mixing**, an m-th order recurrence (window size m):

$$\mathbf{h}_t = \sum_{i=1}^{m} \mathbf{a}_{i,t} \odot \mathbf{h}_{t-i} + \mathbf{a}_{0,t} \odot \mathbf{v}_t, \qquad \text{(H-LRU)}$$

i.e. the same block form with a companion matrix $\mathbf{A}_t^k$ (Figure 1B); $m+1$ selective gates per block instead of $m(m+1)$.

Joint selective gate normalization. Each row of state gates and its input gate are L1-normalized together:

$$\textbf{BD-LRU: } a_{i,j,t} = \frac{f(a'_{i,j,t})}{\sum_{l=0}^{m} |f(a'_{i,l,t})|}; \quad \textbf{H-LRU: } a_{j,t} = \frac{f(a'_{j,t})}{\sum_{l=0}^{m} |f(a'_{l,t})|}. \qquad (1)$$

This guarantees a stable recurrence and a bounded forward pass for any m, and makes the expressivity of the block structure usable by gradient descent.

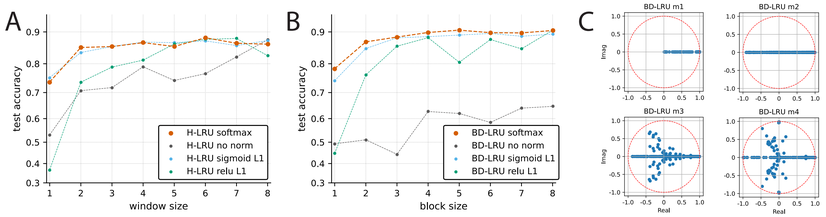


Figure 2: A/B. Accuracy vs. m on compression for gate parametrizations f: with normalization both models scale with m. **C.** BD-LRU eigenvalues on S_5 become complex from $m = 3$.

Permutation Tasks (State Tracking)

Input: a sequence of permutations from S_n; target: their running composition. Larger n needs richer symmetries in the state transition, which **block size m unlocks**, even with **little training data**.

Model	S_3 (10k)	S_3 (250)	S_4 (50k)	S_4 (3k)	S_5 (100k)	Overall
LSTM	**1.000**	0.316	**1.000**	0.368	**1.000**	0.737
Mamba2	0.657	0.281	0.429	0.119	0.261	0.349
DeltaNet[-1,1]	**1.000**	0.264	0.507	0.138	0.140	0.410
DeltaProduct$_4$[-1,1]	**1.000**	0.279	**1.000**	0.136	0.140	0.511
H-LRU m5	0.622	0.318	0.448	0.194	0.377	0.392
BD-LRU m1	0.563	0.381	0.343	0.218	0.215	0.344
BD-LRU m2	**1.000**	0.488	0.699	0.356	0.338	0.576
BD-LRU m3	**1.000**	**1.000**	**1.000**	0.426	0.484	0.782
BD-LRU m4	**1.000**	**1.000**	**1.000**	**1.000**	0.882	0.976
BD-LRU m5	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**

Table 1: Accuracy on permutation composition; column labels give the group S_n and the number of training samples. BD-LRU is the only model that solves every task, also with scarce data.

Language Modeling (FineWeb-Edu)

Block size improves perplexity: moderate block sizes beat the diagonal model and keep improving with hidden size where it saturates. Beyond perplexity we find a **dissociation**: retrieval-oriented models (Mamba2, DeltaNet) win on perplexity and retrieval, while our models are clearly ahead on tasks that require **tracking structure**.

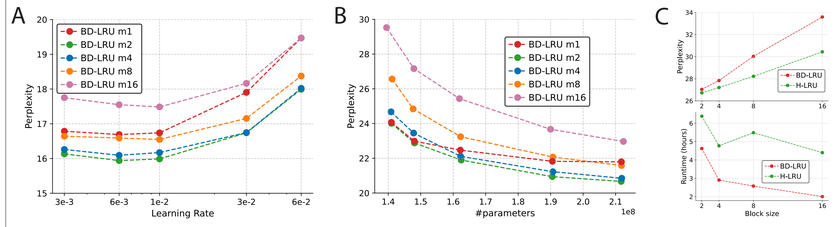


Figure 3: A. Perplexity vs. learning rate (210M parameters, 10B tokens). **B.** Scaling with hidden size (2.5B tokens). **C.** Runtime vs. perplexity (140M parameters).

Key Findings

- **The structure of state mixing drives performance;** joint gate normalization is what lets both models improve with m.
- **Fit the inductive bias to the task:** channel mixing (BD-LRU) for state tracking, temporal mixing (H-LRU) for compression.
- **Language modeling:** block size lowers perplexity; state mixing shows on structure tracking, not on retrieval.
- **Moderate block sizes perform best** and, with our scans, cost about as much as diagonal models in practice.
- **Block size buys expressivity cheaper than Householder rank.**

Block-Diagonal Parallel Scan

A parallel scan [9] evaluates the recurrence in $\log_2 T$ parallel levels by composing steps $\mathbf{c}_t = [\mathbf{A}_t, \mathbf{b}_t]$, $\mathbf{b}_t = \mathbf{a}_{0,t} \odot \mathbf{v}_t$:

$$\mathbf{c}_i \bullet \mathbf{c}_j = [\mathbf{A}_j \mathbf{A}_i,\ \mathbf{A}_j \mathbf{b}_i + \mathbf{b}_j].$$

A dense composition costs H^3; ours stays block-diagonal and costs Nm^3, within an $O(m^2)$ factor of a diagonal scan (cf. [10]).

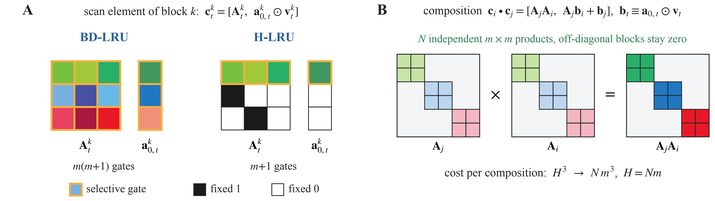


Figure 4: A. Scan element of one block for BD-LRU and H-LRU ($m = 3$). **B.** The composition stays block-diagonal: N small products instead of one $H \times H$ product.

Parallel scans are orders of magnitude faster than the sequential reference; our compiled Triton kernel is the fastest at every sequence length.

Implementation (ms / MB)	T=256	T=1024	T=4096	T=16384	T=65536
Sequential (PyTorch)	78.4 / 178	300 / 461	2383 / 1597	–	–
Block hopscan (PyTorch)	16.9 / 217	20.8 / 618	49.7 / 2219	195 / 8616	776 / 34199
Block hopscan + `compile`	3.2 / 132	3.4 / 227	9.9 / 406	37.4 / 1366	–
Triton scan	3.2 / 208	3.3 / 590	7.1 / 2109	26.7 / 8181	106 / 32456
Triton scan + `compile`	**3.7** / 132	**3.8** / 228	**4.5** / 546	**16.5** / 1932	**64.8** / 7478

Table 2: Training-step time in ms / peak memory in MB for the five scan implementations (BD-LRU, $m = 4$, $B = 4$, ∼1M parameters, H100, bf16); "–": over budget.

Block size vs. Householder rank: at every matched rank BD-LRU is faster and leaner than the DeltaNet family, and the margin grows with T.

T	**Model** (ms / MB)	**rank 1**	**rank 2**	**rank 4**	**rank 8**
4096	BD-LRU (block size m)	**4.0** / **620**	**4.5** / **516**	**4.5** / 546	**6.4** / **1447**
	H-LRU (window size m)	4.1 / 620	5.7 / 790	7.7 / 1333	17.6 / 6678
	DeltaNet / DeltaProduct (H_n)	7.7 / 1203	8.4 / 1646	10.2 / 1983	11.8 / 2204
	LSTM	89.2 / 299	–	–	–
16384	BD-LRU (block size m)	**14.4** / **2228**	**16.3** / **1821**	**16.5** / **1932**	**24.2** / **5539**
	H-LRU (window size m)	14.3 / 2228	20.7 / 2910	29.1 / 5083	69.1 / 26462
	DeltaNet / DeltaProduct (H_n)	22.1 / 4512	33.5 / 6330	43.7 / 7683	49.2 / 8570
	LSTM	346 / 919	–	–	–

Table 3: Cost of expressivity at matched parameters (∼1M, $B = 4$): ms / MB per training step vs. block or window size m (ours) or Householder rank H_n (DeltaNet: $H_n = 1$).

References

[1] A. Orvieto et al. "Resurrecting recurrent neural networks for long sequences".
[2] A. Gu et al. "Mamba: Linear-time sequence modeling with selective state spaces".
[3] W. Merrill et al. "The illusion of state in state-space models".
[4] N. M. Cirone et al. "Theoretical foundations of deep selective state-space models".
[5] S. Hochreiter et al. "Long short-term memory".
[6] T. Dao et al. "Transformers are ssms: Generalized models and efficient algorithms through structured state space duality".
[7] S. Yang et al. "Parallelizing linear transformers with the delta rule over sequence length".
[8] J. Siems et al. "Deltaproduct: Improving state-tracking in linear rnns via householder products".
[9] G. E. Blelloch. "Prefix sums and their applications".
[10] B. Walker et al. "Structured Linear CDEs: Maximally Expressive and Parallel-in-Time Sequence Models".

Code: github.com/IgorDn8/mad-lab & github.com/IgorDn8/plainLM **Contact:** igor@nisys.ai