DO STATE SPACE MODELS NEED SEPARATE B AND C PROJECTIONS? REVISITING B-C TYING FROM S4D TO MAMBA

Anonymous authors
Paper under double-blind review

ABSTRACT

Sequence models keep growing more parameter-hungry, but not every learned projection is necessarily doing distinct work; on memory- and power-constrained hardware, knowing which architectural roles are actually necessary rather than merely conventional is directly actionable. We study *projection redundancy*: whether the distinct linear roles a sequence model uses to read and write information can be collapsed onto shared parameters without losing capability. Diagonal state space models (SSMs) such as S4D, S5, and Mamba read and write their hidden state through two matrices, B (input) and C (output); we test tying $C = B$ across four case studies, multi-seeded throughout except where flagged single-seed. For a static SSM (S4D), tying is a clean free lunch — no measurable cost at any depth or task tested, including a byte-level language model — cutting the SSM layer's parameters by a third. For a selective SSM (Mamba), tying carries a real $\sim$14% capacity cost, traced to real-valued B, C forcing a tied mode's contribution to be non-negative (unlike S4D's complex-valued state); a fixed sign mask fixes this at zero extra parameters, recovering $\sim$86% of the gap, and a parameter-matched recursive check confirms the fix is a genuine effect rather than fewer parameters in disguise, beating a matched-budget free model decisively on a real-corpus language model. A plain RNN with a genuine nonlinearity inside the recurrence again costs nothing reproducible, and a systematic study of attention found tying two of three Q, K, V projections costs $< 1\%$ relative accuracy (Kayyam et al., 2026). Post hoc, converting an already-trained free model to tied is exact for S4D, damaging but fine-tune-recoverable for Mamba, and on a real language model, over 80% recoverable (single seed) using only the frozen model as a synthetic teacher, without the original training data. Together, these four case studies show projection redundancy is architecture-dependent in its mechanics, but when tying costs something, the cause is often diagnosable and cheaply fixable, suggesting that checking whether a read/write projection pair is actually necessary is a cheap, broadly applicable audit for future sequence-model designs, not a peculiarity of any one of the architectures tested here.

1 INTRODUCTION

Modern sequence models increasingly separate the roles of *writing* information into a representation from *reading* it back out: three learned projections in self-attention (Q, K, V), two in linear state space models (SSMs), the input matrix B and the output matrix C. We call the question of how much of this separation is actually necessary — whether tying two projections together loses real capability, or whether gradient descent simply finds an equally good solution in the smaller parameter space — *projection redundancy*. The answer matters beyond parameter counting: on memory- and power-constrained edge hardware, where every stored weight and every read/write access carries a direct energy and latency cost, collapsing a redundant projection is one of the few compression levers that need not trade away accuracy, unlike quantization or pruning.

We test $C = B$ tying (or the analogous projections) on three architectures — a static diagonal SSM (S4D), a selective SSM (Mamba), and a plain nonlinear RNN — and compare against a fourth case study, transformer attention, studied separately in Kayyam et al. (2026). For S4D, tying is close

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

to free at every scale we test, from a single linear layer to a real-corpus byte-level language model (Section 4.1), cutting the SSM layer's parameters by a third throughout. Mamba disagrees: tying carries a real capacity cost (Section 4.2), which we trace to a precise algebraic cause — B_t, C_t being real-valued forces a tied mode's contribution to be non-negative, unlike S4D's complex-valued state — and fix at zero extra parameters with a fixed sign mask (Section 4.3), recovering most of the gap. A recursive check against parameter-matched free models confirms the fix is doing something real, not merely reducing parameter count, most decisively on a real-corpus language model (Section 4.3). A plain RNN with a genuine nonlinearity *inside* the recurrence again costs nothing reproducible (Section 6), ruling out linearity as the explanation for the S4D result. We further ask whether tying can be applied *post hoc* to an already-trained free model (Section 5): exact for S4D, damaging but fine-tune-recoverable for Mamba, and on a real language model, mostly recoverable even without the original training data.

This paper's contribution is diagnostic rather than purely empirical: rather than reporting "tying is free" or "tying costs $X\%$" per architecture, we identify *why* the two SSM architectures disagree and show the disagreement is fixable once diagnosed. Section 2 situates this against weight-tying and parameter-sharing work; Section 3 gives the model and the algebraic argument; Section 4 reports the SSM case studies; Section 5 covers post-hoc conversion; Section 6 synthesizes across all four architectures; Sections 7–8 close.

2 Related Work

Weight tying and parameter sharing. Tying parameters across ostensibly distinct roles has a long history as a cheap efficiency win. Press & Wolf (2017) tie a language model's input and output embedding matrices, reducing model size with no loss in perplexity. Lan et al. (2020) (ALBERT) share parameters across all transformer layers, and Dehghani et al. (2019) (Universal Transformers) share a single block's weights across depth, both trading a large parameter reduction for little or no quality loss on their respective tasks. These tie parameters across *layers*; our question is narrower and orthogonal: we tie two projections *within* a single layer that play nominally different read/write roles.

Sharing read/write projections in attention. The most direct precedent is multi-query (Shazeer, 2019) and grouped-query (Ainslie et al., 2023) attention, which share key/value projections across (groups of) heads, primarily to reduce the KV-cache memory-bandwidth bottleneck at autoregressive inference time rather than to answer an expressivity question. Kayyam et al. (2026) measures the accuracy cost of these and related Q, K, V tying schemes directly (Table 11); this paper asks the same question for SSMs' B, C and, unlike that study, identifies a precise algebraic reason (Section 4.2) for *when* tying is and is not free, rather than only an empirical accuracy comparison.

Parameter-efficient reparameterization more broadly. Hu et al. (2021) reparameterize weight *updates* as low-rank matrices for efficient fine-tuning, a different mechanism from this paper's literal parameter tying but the same underlying bet that a model's effective parameterization is smaller than its nominal one. Frankle & Carbin (2019)'s lottery ticket hypothesis makes this bet explicit at the level of individual weights via pruning rather than architectural roles; our results instead identify, for one specific architectural role (the SSM's read/write projections), exactly when this kind of redundancy is present and why.

State space model architectures. The SSM variants this paper studies span a design lineage rather than a single fixed recipe: S4 (Gu et al., 2022a) models a linear time-invariant recurrence through a structured (diagonal-plus-low-rank) state matrix, computed as a global convolution to avoid an explicit sequential scan during training; S4D (Gu et al., 2022b) and DSS (Gupta et al., 2022) simplify this to a fully diagonal state matrix at no measurable quality cost; S5 (Smith et al., 2023) reformulates the same recurrence per-timestep rather than as a convolution; and Mamba (Gu & Dao, 2023) makes the recurrence itself *input-dependent* (a "selective" scan), trading the closed convolutional form for data-dependent routing. Each of these papers optimizes *how* the recurrence is computed and parameterized. This paper instead holds the recurrence's own design fixed within each architecture and asks a narrower, largely orthogonal question: whether the input (B) and output (C) projections that read from and write to that recurrence are both actually necessary. Framed this way, the S4D-versus-Mamba comparison in this paper is not a comparison of architectures but of how one specific design axis — real- versus complex-valued state, static versus selective — changes the answer to that same question.

Real- vs. complex-valued recurrences. Most relevant to our mechanistic account of *why* tying costs something in Mamba but not S4D (Section 4.2) is Orvieto et al. (2023), who show a carefully parameterized *real*-valued linear recurrence (the Linear Recurrent Unit) can match complex-valued deep SSMs on long-range tasks given the right initialization and normalization scheme. Their result and ours are complementary: they show real-valued recurrences need not underperform complex-valued ones *in general*, while we show one specific consequence of realness — the loss of sign freedom under $B = C$ tying — that does measurably cost something and that has a cheap, targeted fix (Section 4.3), rather than requiring the fully general reparameterization their work introduces.

3 Background and Method

Diagonal SSMs. S4D (Gu et al., 2022b), S5 (Smith et al., 2023), and the linear recurrence at the core of Mamba (Gu & Dao, 2023) all parameterize a linear time-invariant system with diagonal (complex) state transition, $h_t = \bar{A}h_{t-1} + \bar{B}u_t$, $y_t = \mathrm{Re}(Ch_t)$. Because the recurrence is linear, S4D/S5 admit an equivalent convolutional form, avoiding an explicit sequential scan during training (Gu et al., 2022a). Mamba departs by making B_t, C_t, and the step size Δ_t all *input-dependent* per timestep (with A remaining static and *real*-valued): this removes the closed form, requiring a genuine (parallel-scan) recurrence at both training and inference. This static/complex vs. selective/real distinction turns out to matter a great deal for whether tying $C_t = B_t$ is free (Section 4.2).

Model and tying. We use S4D-Lin initialization ($A_n = -\frac{1}{2} + i\pi n$) with complex learnable $B, C \in \mathbb{C}^{D\times N}$: *free* keeps them independent parameters; *tied* sets $C = B$ literally (the same tensor, not merely the same initialization), removing exactly one third of the SSM layer's parameters (A, Δ unaffected). For Mamba, B_t, C_t are computed per-timestep via a shared input projection in *free* mode, or one shared projection (so $B_t = C_t$ at every timestep) in *tied* mode.

Why tying might be free. In a single-input single-output (SISO) diagonal channel with fixed A, the impulse response depends only on per-mode gains $g_n = B_n C_n$; any two parameterizations realizing the same $\{g_n\}$ realize the same input-output map, so tying is lossless for this model class exactly when it does not shrink the set of gains a single mode can realize.

Proposition 1. *Let $g \in \mathbb{C}$. There exists $B \in \mathbb{C}$ with $B^2 = g$. Consequently, the map $(B, C) \mapsto BC$ over independent $B, C \in \mathbb{C}$ and the map $B \mapsto B^2$ (the tied case, $C = B$) have the same image, $\mathbb{C}$.*

Proof. If $g = 0$, take $B = 0$. Otherwise $g = re^{i\theta}$, $r > 0$: $B = \sqrt{r}\, e^{i\theta/2}$ satisfies $B^2 = g$. The free parameterization trivially achieves every g (fix $C = 1$, $B = g$). □

Expressivity of the diagonal SISO transfer function is therefore unchanged by tying *in principle*; whether gradient descent finds this solution in practice is what we test. The corresponding statement fails over the reals:

Corollary 1. *Let $g \in \mathbb{R}$. If $g < 0$, no $B \in \mathbb{R}$ satisfies $B^2 = g$. Hence, over $\mathbb{R}$, $(B, C) \mapsto BC$ has image $\mathbb{R}$, while the tied map $B \mapsto B^2$ has image $\mathbb{R}_{\geq 0} \subsetneq \mathbb{R}$: tying strictly shrinks the achievable per-mode gains whenever B, C are real.*

Proof. $B^2 \geq 0$ for all real B. Free achieves any $g \in \mathbb{R}$ via $C = 1, B = g$. □

This is exactly what makes Mamba's real-valued A, B_t, C_t behave differently (Section 4.2).

Architectures and training. We test three settings of increasing realism per SSM type: a single linear SSM layer mapping a scalar input to D channels; the same task through a stacked nonlinear block (LayerNorm → depthwise diagonal SSM → Linear → GLU → residual, 1–3 deep, each tying $C = B$ independently); and real tasks (sequential MNIST, byte-level language modeling). All models train with Adam. Early single-seed comparisons proved unreliable in both directions during development (a small apparent cost that turned out to be noise in one case, a near-tie that hid a real cost in another), so every result below is multi-seeded (3–5 seeds) unless flagged otherwise. Exact hyperparameters per experiment and the memory-capacity metric's exact definition are in Appendix A. Code and raw logs are available alongside this paper.

4 EXPERIMENTS

4.1 S4D: TYING IS FREE AT EVERY SCALE TESTED

We compare free and tied on: a single linear SSM layer (memory capacity, easy and hard/undercapacity regimes); the same task through a 1–3 layer stacked nonlinear block; sequential MNIST (784-step pixel-by-pixel scan, 3-layer stack); and byte-level ($vocab$=256) language modeling on enwik8 (90M/5M/5M split, d_{model}=256, d_{state}=64, 6 layers, T=1024, 30k steps). Table 1 summarizes; full per-setting tables and training curves are in Appendix B.

Table 1: S4D: free vs. tied across every setting tested (mean ± std over 3–5 seeds where multi-seeded; single-layer regimes use 5-seed point estimates, see Appendix B). Gap/SE values are as explicitly reported for that setting; "n.s." means the gap was smaller than the configurations' own seed-to-seed variation, with no formal ratio computed.

Setting	Free	Tied	Gap/SE
Single layer, easy (32 lags)	31.997 ± 0.001	31.997 ± 0.002	n.s.
Single layer, hard (128 lags)	8.048 ± 0.034	8.047 ± 0.026	n.s.
Stacked, 1–3 layers (capacity)	26.0–32.1	26.2–32.1	-0.5 to 0.1
Sequential MNIST (final acc.)	$99.15\% \pm 0.10$	$99.01\% \pm 0.20$	n.s.
enwik8 LM (val. bpb, 3 seeds)	1.8800 ± 0.0016	1.8824 ± 0.0016	≈ 1.5

Free and tied are statistically indistinguishable everywhere we tested S4D, including a single isolated layer and a real-corpus language model, with a consistent ∼33% SSM-parameter saving throughout (the sMNIST classifier's *total-model* saving is diluted to 19.7% by its non-SSM readout layers; see Appendix H for why this dilution happens and how large it is at other configurations). This closes the question at real-corpus LM scale, not only on synthetic probes: tying $C = B$ costs nothing measurable for a static diagonal SSM at any scale or task tested here. Figure 1 shows why the hard-regime row is a match and not merely a coincidence of totals: the full per-lag capacity curve traces out essentially the same decay shape for both configurations, not just the same aggregate sum.

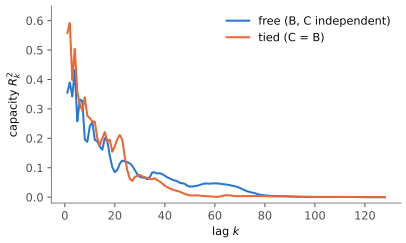


Figure 1: Per-lag memory capacity, single linear SSM layer, hard (undercapacity) regime. Both configurations trace out essentially the same decay shape.

4.2 MAMBA: A REAL COST, DIAGNOSED

Mamba computes B_t, C_t, Δ_t per-timestep from the input via linear projections, with A static and real-valued; the free/tied distinction carries over directly. Using a single Mamba block (d_{model}=16, d_{inner}=32, d_{state}=8, 24 lags), 5 seeds: free averages 6.819 ± 0.603, tied 5.872 ± 0.210 — free ∼14% higher, gap/SE $\approx$ 3.3. **Unlike every S4D result, tying has a real, consistent cost here.** A 25-seed extension (motivated by an unrelated post-hoc check that surfaced free-training instability at this budget; full numbers in Appendix C) confirms this on 30 pooled seeds at gap/SE = 5.11, and reveals free has a real ∼1-in-12 chance of catastrophic optimization failure that tied is immune to — **tying appears to make training more reliable, not merely more parameter-efficient** — though we have not characterized this failure mode beyond this one synthetic task.

Why S4D and Mamba disagree. Corollary 1 applies directly to Mamba's real-valued A, B_t, C_t: tying gives $C_t \bar{B}_t = \Delta_t B_t^2 \geq 0$ at every timestep (the strictly-positive step size $\Delta_t = \text{softplus}(\cdot)$ rescales this coupling but can never flip its sign), so a tied channel can no longer represent a *negative* same-timestep input-output coupling. This is a structural restriction, not an optimization artifact, and is the natural, formally grounded explanation for why S4D's complex state (Proposition 1) shows no cost while Mamba's real one (Corollary 1) does.

4.3 A ZERO-PARAMETER FIX

If lost sign freedom is the cause, restoring it without giving back tying's parameter saving should recover the cost. We test **tied_sign**: $C_t = B_t \cdot s$ for a *fixed* random ± 1 mask s of the same shape, stored as a non-learned buffer (zero extra parameters over plain tied); and, as a cautionary comparison, **tied_gain**: a *learned* low-rank sign/scale ($d_{\text{inner}} + d_{\text{state}}$ extra parameters).

Table 2: Mamba memory capacity, gain-fix comparison, 5 seeds.

Config	SSM params	Total capacity (mean $\pm$ std)
Free	1,088	6.819 ± 0.603
Tied	832	5.872 ± 0.210
Tied + learned low-rank gain	872 (+40)	6.101 ± 1.248
Tied + fixed random sign (tied_sign)	832 (+0)	6.688 ± 0.200
Tied + fixed alternating sign (tied_sign_alt)	832 (+0)	6.939 ± 0.055

tied_sign closes ∼86% of the tied-to-free gap at zero extra parameters and is statistically indistinguishable from free (gap/SE $\approx$ 0.46); the *deterministic* alternating mask, **tied_sign_alt** ($s_n = (-1)^n$), does slightly better still, with $3.6\times$ tighter variance — we report this as an empirical finding, without a mechanistic account of why determinism itself helps. tied_gain is a cautionary result in the opposite direction: it improves on plain tied on average but far less reliably (one seed collapses to 3.615, worse than tied's worst run) — *a fixed structural fix beat a learned one here*.

On sequential MNIST (single block, 3 seeds, matched seeding), tied_sign leads both free and tied on both metrics across all three seeds, recovering the full final-epoch gap and then some:

Table 3: Mamba sequential MNIST, single block, 3 seeds (mean $\pm$ std).

Config	SSM params	Final-epoch acc.	Best-epoch acc.
Free	4,224	$89.28\% \pm 0.65$	$89.83\% \pm 0.84$
Tied	3,200 (-24.2%)	$89.23\% \pm 1.56$	$89.97\% \pm 0.53$
Tied + fixed sign	3,200 (+0)	$89.83\% \pm 0.70$	$90.70\% \pm 0.41$

tied_sign matching or beating free is confirmed here (a secondary, hardware-dependent comparison is in Appendix C).

Does stacking rescue Mamba's tying cost, the way S4D's illusory single-layer tax vanished once stacked (Section 4.1)? We rerun the memory-capacity task at 1, 2, and 3 Mamba blocks, for free, tied, and tied_sign, 5 seeds each.

Table 4: Stacked Mamba memory capacity, free/tied/tied_sign, mean $\pm$ std over 5 seeds.

Layers	Free	Tied	Tied + fixed sign
1	6.908 ± 0.511	5.872 ± 0.277	6.649 ± 0.185
2	9.570 ± 0.635	8.743 ± 0.106	10.147 ± 0.332
3	12.311 ± 0.393	11.039 ± 0.238	12.684 ± 0.377

The free-vs-tied gap/SE is 3.99, 2.87, and 6.19 at $L{=}1, 2, 3$: **significant at every depth with no trend toward vanishing** — unlike S4D, where the apparent single-layer cost was never a real, depth-dependent effect to begin with. The free-vs-tied_sign gap/SE, by contrast, is 1.07, -1.80, and -1.53 — **not significant at any depth**, so the fix's robustness is not an artifact of testing only one block.

Is the fix just fewer parameters in disguise? We test this with a recursive width-matched procedure: train tied_sign at some width d_{inner}; find the closest-matching *free* model by narrowing d_{inner} alone to the same SSM parameter count; compare, multi-seeded; if not clearly worse, tie that narrower model too and repeat. Run on three tasks — memory capacity, sMNIST, and a small

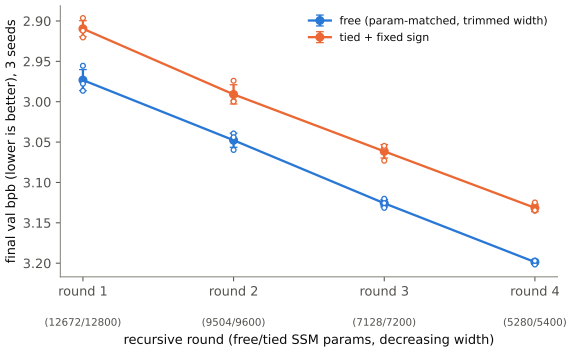


Figure 2: Four rounds of the recursive width-matched check, SlimPajama byte-level LM (small, properly-powered configuration), 3 seeds per condition per round. SSM parameter counts (free/tied) are annotated per round; width (d_{inner}) decreases left to right: free $48\to36\to27\to20$, tied_sign $64\to48\to36\to27$. Lower bpb is better; the y-axis is inverted so "up" means better on this plot too.

SlimPajama byte-level LM — the LM version is the cleanest and most decisive of the three; full detail on all three, including an abandoned memory-constrained LM attempt that did not replicate once properly powered, is in Appendix E.

tied_sign is significantly better than the width-matched free model at every single round (gap/SE -5.49, -5.44, -9.62, -18.40), and the gap grows more decisive as width narrows — the opposite of noise washing out a real effect. Memory capacity and sMNIST point the same direction (tied_sign never significantly beaten by a naive shrink) but as a moderate and a statistically inconclusive result respectively, rather than a clean win (Appendix E); we lean on the language-model result as this section's primary evidence. This comparison is specific to inner-width reduction and does not establish optimality among all budget-matched architectures.

4.4 Language modeling at scale

The Mamba analogue of the S4D enwik8 check ($d_{\text{model}}=128$, $d_{\text{inner}}=64$, $d_{\text{state}}=16$, 4 layers, $T=1024$, 30k steps $\times$ 3 modes, 3 seeds):

Table 5: Mamba enwik8 language modeling, 3 seeds (mean $\pm$ std).

Config	Final val. bits-per-byte	SSM params
Free	2.1325 ± 0.0036	16,896
Tied	2.1411 ± 0.0011	12,800 (-24.2%)
Tied + fixed sign	2.1381 ± 0.0008	12,800 ($+0$ vs. tied)

The free-tied gap (0.0086 bpb, $\sim$0.40% relative) is far smaller than the memory-capacity gap, with gap/SE ≈ 3.5 — consistent in direction across all three seeds but not conventionally significant at this seed count; tied_sign closes 54–73% of it. A single-seed check at $\sim$29.3M ($160\times$ this configuration's parameter count) shows the same order of magnitude (0.45% relative free-tied gap). A $\sim$150M point ($820\times$) gets the same 3-seed treatment as the enwik8 result above, rather than resting on a single seed:

Table 6: Mamba SlimPajama byte-level LM, $\sim$150M parameters, 3 seeds (mean $\pm$ std).

Config	Final val. bits-per-byte	SSM params
Free	2.0109 ± 0.0101	8,094,720
Tied	2.0048 ± 0.0058	7,398,400 (-8.6%)
Tied + fixed sign	2.0076 ± 0.0120	7,398,400 (-8.6%)

Both tied and tied_sign land nominally *ahead* of free here, with gaps smaller than the seed-to-seed std — the same sign flip as a further single-seed check at $\sim$300M ($1{,}600\times$, Appendix I: tied nominally 0.80% ahead of free). **At every real-corpus LM scale tested, the free-vs-tied gap stays an order of magnitude smaller than the memory-capacity gap and is not consistently signed**, unlike the depth-independent capacity cost above.

5 Post-hoc tying: recovery via fine-tuning

A more practically motivated question than training free and tied from scratch side by side: can an already-trained *free* model be converted to tied *after* training, without retraining from scratch, and

does recovering any lost performance require the original training data — relevant whenever only a checkpoint is available, not the training corpus?

For S4D, Proposition 1 is directly constructive: since the S4D kernel uses B, C only through $C\bar{B} \propto CB$, setting $B_n^{\text{new}} = C_n^{\text{new}} = \sqrt{B_n C_n}$ (elementwise complex square root) reproduces the identical kernel with zero fine-tuning (verified numerically: maximum output difference 1.3×10^{-7}, at float32 noise floor). **For S4D, post-hoc tying is a free, closed-form, lossless conversion.**

For Mamba, no closed form exists (B_t, C_t interact through a time-varying selective scan); we instead merge each block's B/C rows (averaging by default) and fine-tune every weight jointly.

Table 7: Post-hoc conversion and recovery, Mamba (memory capacity, 5 seeds; enwik8, single seed).

Condition	Memory capacity	enwik8 val. bpb
Free (just-trained)	6.819 ± 0.603	2.1378
Zero-shot tied (merge = average)	3.194 ± 0.599	2.7157
Tied, fine-tuned (real data)	6.771 ± 0.301	2.1377
Tied, fine-tuned (data-free, distillation only)	—	2.2491

Zero-shot damage is substantial ($\sim$53% relative capacity drop; $+0.578$ bpb on enwik8), but ordinary fine-tuning at a sufficient step budget fully recovers it (memory-capacity gap/SE ≈ 0.16; enwik8 essentially exact). An initially damaging shorter-budget gap traced to a *co-adaptation mismatch* — every other weight stays co-adapted to the free model's unconstrained dynamics — confirmed by two independent fixes (resetting co-adapted weights; simply extending the fine-tuning budget) and ruled out as a step-size problem (higher learning rates reliably backfire); full diagnostic detail in Appendix G. On enwik8, fine-tuning *without* the original corpus at all — using only the frozen free model as a distillation teacher on random synthetic bytes, matching its output distribution via KL divergence — still **recovers 80.7% of the zero-shot gap** (2.2491 bpb), single seed given cost but with an effect size large enough that this is unlikely to be a single-seed artifact. The zero-parameter sign fix transfers to this setting only partially: it prevents catastrophic zero-shot damage from an unlucky random mask choice (Appendix G), but its effect on the fine-tuning recovery ceiling specifically remains unresolved at $n = 5$ — a longer fine-tuning budget reversed rather than settled the ranking between the random and deterministic mask variants:

Table 8: Sign-fix recovery ranking at a shorter vs. longer fine-tuning budget, memory capacity, 5 seeds (mean $\pm$ std).

Condition	2000 steps	6000 steps
Plain tied	6.243 ± 0.454	6.771 ± 0.301
Tied_sign (random mask)	5.863 ± 1.023	7.221 ± 0.592
Tied_sign_alt (deterministic mask)	6.185 ± 1.588	5.521 ± 1.617

At 2000 steps the deterministic mask nominally leads and the random mask trails even plain tied; at 6000 steps the ordering inverts. None of the pairwise differences reach significance at $n = 5$ at either budget — **the reversal itself, not either individual ordering, is the finding**, underscoring this paper's broader insistence on multi-seed comparisons before trusting any single result. Full co-adaptation and learning-rate diagnostics behind the 2000-vs-6000-step budget choice are in Appendix G.

6 Discussion: projection redundancy across architectures

To test whether the S4D free lunch depends on the recurrence itself being linear, we ran the identical tying question on a plain (Elman) RNN, $h_t = \tanh(W_{hh} h_{t-1} + W_{ih} \odot u_t)$, $y_t = \sum_n (h_t \odot W_{ho})_n$, tying $W_{ho} = W_{ih}$ — a genuine nonlinearity *inside* the recurrence, not only between stacked blocks. Across a single-layer synthetic probe, a 1–3 layer stacked version, and sequential MNIST (all multi-seeded), **tying shows no reproducible cost at any depth or on the real task**, at a far smaller parameter saving (5–9%, since the untied dense W_{hh} dominates per-channel parameter count) than

S4D's one-third, and with substantially noisier training throughout. This rules out "the recurrence is linear" as an explanation for the S4D free lunch.

Table 9: Vanilla RNN memory capacity, single layer (point estimate) and stacked 1–3 layers (mean $\pm$ std, 3 seeds).

Setting	Free	Tied	Params (tied vs. free)
Single layer, easy (2 lags)	1.999 / 2	1.996 / 2	640 vs. 704 (-9.1%)
Single layer, hard (32 lags)	8.024 / 32	7.908 / 32	2,304 vs. 2,432 (-5.3%)
Stacked, 1 layer	7.024 ± 0.81	7.021 ± 0.006	640 vs. 704 (-9.1%)
Stacked, 2 layers	8.024 ± 0.01	8.025 ± 0.01	1,280 vs. 1,408 (-9.1%)
Stacked, 3 layers	8.020 ± 0.00	8.033 ± 0.01	1,920 vs. 2,112 (-9.1%)

Full detail — including a training-stability side-finding (free's single-layer seed-to-seed variance is $\sim$100$\times$ tied's) and an sMNIST comparison whose nominal direction reversed under a $3\times$ larger seed count — is in Appendix D.

Attention, SSMs, and this RNN disagree on how many projections are redundant (one of three vs. one of two) and on how large the practical savings are: 5–9% for the RNN's dense-recurrence case; a flat one-third for S4D and QKV, since both tie exactly one of a fixed small set of equal-sized projections; and a more config-dependent 6.6–33% for Mamba, shrinking as the model widens relative to its fixed state dimension. **All of these figures are the read/write layer's own parameter savings, not a deployed model's total parameter count** — surrounding non-SSM projections dilute the total-model savings substantially, and by a very different amount for the two SSM families:

Table 10: Parameter accounting across every real-corpus LM configuration in this paper (free mode). "C-only" is exactly the parameter count removed by tying; full per-block formulas in Appendix H.

Configuration	Total params	SSM params	SSM/total	C-only	Saving/total
Mamba, enwik8 /SlimPajama-small (d=128)	183,552	16,896	9.2%	4,096	2.23%
Mamba, SlimPajama (d=640)	29,277,696	1,830,400	6.3%	225,280	0.77%
S4D, enwik8 (d=256)	1,515,776	591,360	39.0%	196,608	13.0%

This is explained entirely by one ratio, $d_{\text{state}}/d_{\text{model}}$: Mamba keeps $d_{\text{state}} = 16$ fixed while d_{model} grows, so B, C (which scale as $O(d_{\text{inner}} d_{\text{state}})$) become an ever-smaller slice of a model dominated by `in_proj`/`out_proj` (which scale as $O(d_{\text{model}} d_{\text{inner}})$), shrinking the total-model saving from 2.23% to 0.77% as d_{model} goes from 128 to 640, and to well under 1% at typical Mamba hyperparameter ratios more broadly. S4D's $d_{\text{state}} = 64$ against $d_{\text{model}} = 256$ is a much larger ratio, keeping a real, double-digit total-model saving (13.0%). **Tying's total-parameter value is therefore architecture- and hyperparameter-dependent, not a fixed percentage: largest for static SSMs with a state dimension that is a substantial fraction of model width, smallest for selective SSMs at typical Mamba ratios** — so the abstract's "-33% for S4D" and Table 5's "-24.2% for Mamba" should be read as SSM-layer savings throughout this paper, not as the fraction of a full deployed model those numbers shrink. The four architectures agree, though, on the qualitative shape: tying *some* read/write projections together is close to free, while collapsing the read/write distinction more aggressively ($Q{=}K{=}V$) costs substantially more (Kayyam et al., 2026), full detail in Appendix F:

Table 11: Synthetic-task token accuracy, QKV projection-sharing variants (Kayyam et al., 2026), averaged over a config sweep.

Variant	Avg. accuracy
QKV (baseline, all free)	0.758
Q$\neq$K=V (tie two of three)	0.751
Q=K=V (tie all three)	0.651

Projection redundancy is therefore architecture-dependent in its mechanics — what fails, and why, differs by architecture (real- vs. complex-valued state for the SSMs; degree of collapse for attention) — but when tying does cost something, the cause has so far always been diagnosable and cheaply fixable, and some degree of read/write sharing has been available in every architecture tested here.

This suggests projection redundancy is a property of the read/write *roles* themselves — reading and writing information through nominally separate channels — rather than of any one architecture's specific mechanics, and that an architecture-level finding like "$C{=}B$ costs nothing" is not automatically portable to a superficially similar architecture without checking whether the underlying algebra still permits it.

7 LIMITATIONS

Both language models remain modest scale ($\sim$1.5M/$\sim$180K parameters for S4D/Mamba) relative to the QKV study's 300M/1.2B scale, and no long-range benchmark such as Long Range Arena has been tried; three further Mamba web-text checks (Section 4.4) extend to $\sim$160–1,600$\times$ this scale, but only the 150M point is properly multiseeded. Every SSM result here uses S4D or Mamba specifically, not a third architecture such as S5. The recursive parameter-matching checks (Section 4.3) are unevenly powered across tasks — moderate on memory capacity, inconclusive on sMNIST, decisive only on the language model — and should not be read as three equally strong replications. The post-hoc study's flagship enwik8 recovery result is single seed, though its effect size (80%+ data-free recovery) is large; the sign-fix recovery comparison specifically remains unresolved at $n = 5$. The underlying free-vs-tied sMNIST gap is unsettled between two 3-seed studies on different GPU hardware, though tied_sign matching or beating free holds on both. Proposition 1 and Corollary 1 formally cover only a single SISO diagonal channel; the stacked, nonlinear, and classification results are validated empirically, not proven. The RNN results (Section 6) are noisier and less multiseeded than the SSM results, capped by vanishing gradients rather than by tying, and its sMNIST comparison's direction reversed between $n = 3$ and $n = 10$ (Appendix D).

8 CONCLUSION

For S4D-style diagonal SSMs, tying $C = B$ is a clean free lunch: no measurable capacity or accuracy cost at any depth tested, from a single linear layer to a real-corpus byte-level language model, with a consistent one-third reduction in the SSM layer's parameter count throughout. For Mamba's selective SSM, tying carries a real, depth-independent $\sim$14% capacity cost and a smaller, directionally consistent sMNIST cost; we traced the capacity cost to real-valued B_t, C_t restricting a tied mode's contribution to be non-negative (unlike S4D's complex-valued state) and fixed it at zero extra parameters with a fixed sign mask, closing $\sim$86% of the capacity gap and matching or beating free on sMNIST across two independent seed sets. A recursive parameter-matched comparison confirms this is not merely a parameter-count effect: tied_sign is never significantly beaten by a naive width-matched shrink on memory capacity or sMNIST, and beats one decisively and increasingly on a real-corpus language model. At real-corpus LM scale specifically — enwik8 and three SlimPajama scaling checks from 29.3M to 300M parameters — the free-vs-tied gap stays an order of magnitude smaller than the capacity gap and is not consistently signed, unlike the depth-independent capacity result.

Beyond training tied from scratch, post-hoc conversion is exact and free for S4D, substantially more damaging than training tied from scratch for Mamba (traced to a co-adaptation mismatch, not left unexplained), but fully recoverable by ordinary fine-tuning — and, on a real language model, mostly recoverable (80.7%) even without the original training corpus, using only the frozen model as a synthetic teacher. A third case study, a plain RNN with a genuine nonlinearity inside the recurrence, again costs nothing reproducible, ruling out linear recurrence as the explanation for the S4D result; together with the attention comparison in Kayyam et al. (2026), all four case studies point toward projection redundancy being a property of the read/write *roles* themselves rather than of any one architecture's specific mechanics — architecture-dependent in its particulars, but when a cost exists, diagnosable and cheaply fixable, and available widely enough to be worth checking for in any new sequence model. Practically, this is especially relevant to memory- and power-constrained edge hardware: tying cuts the SSM layer's own parameter count at no accuracy cost for S4D, or recovers almost all of a real cost at zero added parameters for Mamba via the sign-mask fix, without the accuracy trade-off that comes with quantization or pruning. The post-hoc result speaks directly to the common deployment scenario this saving is usually needed for: compressing a checkpoint whose original training data is unavailable, where tying is recoverable by ordinary fine-tuning and, even without that, mostly recoverable using only the frozen model itself as a synthetic teacher.

AI USE STATEMENT

Generative AI tools (Claude, Anthropic) were used to help write and debug the experiment and analysis code (data pipelines, training scripts, and result-aggregation scripts) that produced the numerical results reported in this paper. All AI-assisted code was reviewed, run, and verified by the author before any result was reported. Generative AI tools were not used to generate the paper's text, analysis, or conclusions; the writing, interpretation, and framing of results in this paper are the author's own. The author takes responsibility for the final content of this work, including all claims and artifacts.

REPRODUCIBILITY STATEMENT

Code and raw training/evaluation logs for every experiment reported are available alongside this paper (Section 3). The memory-capacity metric's exact estimator and masking convention are given in Appendix A; per-block parameter formulas and exact counts for every real-corpus configuration used in this paper are in Appendix H. Every architecture (S4D, Mamba, RNN block) reuses one shared, correctness-verified implementation across every experiment that uses it, checked against a hand-unrolled recurrence and a brute-force sequential loop on a non-power-of-two length before any result was trusted.

ETHICS STATEMENT

This work uses only synthetic data and a standard, publicly available text corpus (enwik8/SlimPajama byte streams); it does not involve human subjects, personal data, or applications with foreseeable direct harms. We do not identify ethical concerns specific to this work.

REFERENCES

Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. GQA: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, Ł. Universal transformers. *International Conference on Learning Representations (ICLR)*, 2019.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations (ICLR)*, 2019.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *International Conference on Learning Representations (ICLR)*, 2022.

Gu, A., Gupta, A., Goel, K., and Ré, C. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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

Gupta, A., Gu, A., and Berant, J. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks. *GMD Technical Report 148*, German National Research Center for Information Technology, 2001.

Kayyam, A., Madan Gopal, A., and Lewis, M. A. Do transformers need three projections? A systematic study of QKV variants. *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, PMLR 306, 2026.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. ALBERT: A lite BERT for self-supervised learning of language representations. *International Conference on Learning Representations (ICLR)*, 2020.

Orvieto, A., Smith, S. L., Gu, A., Fernando, A., Gülçehre, Ç., Pascanu, R., and De, S. Resurrecting recurrent neural networks for long sequences. *International Conference on Machine Learning (ICML)*, 2023.

Press, O. and Wolf, L. Using the output embedding to improve language models. *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2017.

Shazeer, N. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Smith, J. T. H., Warrington, A., and Linderman, S. W. Simplified state space layers for sequence modeling. *International Conference on Learning Representations (ICLR)*, 2023.

A MEMORY CAPACITY: EXACT DEFINITION

"Capacity" names several related but not-identical quantities in the reservoir-computing and SSM literature, so this appendix gives the exact definition used everywhere it is reported in this paper, computed identically across every experiment.

Task. The model observes an i.i.d. sequence $u(t) \sim \mathrm{Uniform}(-1, 1)$, $t = 0, \ldots, T-1$. At every timestep t, alongside its main output, the model produces a separate scalar prediction $\hat{y}_k(t)$ for each lag $k \in \{1, \ldots, K\}$, trained to reconstruct the input from k steps earlier, $u(t-k)$.

Per-lag capacity. Restricted to positions $t \geq k$ (earlier positions, where the target is undefined given the sequence start, are masked out of both training and evaluation), per-lag capacity is the squared Pearson correlation coefficient between prediction and target:

$$\mathrm{MC}_k = \frac{\mathrm{Cov}(\hat{y}_k,\, u_{-k})^2}{\mathrm{Var}(\hat{y}_k)\,\mathrm{Var}(u_{-k})},$$

estimated over a held-out batch of sequences (batch size given per experiment in the main text; 1,024–2,048 throughout). $\mathrm{MC}_k = 1$ exactly when $\hat{y}_k$ is an invertible affine function of $u(t-k)$; $\mathrm{MC}_k = 0$ when $\hat{y}_k$ carries no linear information about it. If the empirical variance of the target at lag k falls below 10^{-8} (avoided by construction in every experiment here, since T always exceeds every lag tested by a wide margin), we set $\mathrm{MC}_k = 0$ by convention rather than dividing by a near-zero denominator.

Total capacity. $\mathrm{MC} = \sum_{k=1}^{K} \mathrm{MC}_k$, summed over the full lag range tested in each experiment (given as "K lags" in the main text). This is the single number reported throughout (e.g. "6.819 ± 0.603").

Why this is the right diagnostic here. For a linear time-invariant system with an N-dimensional real state — S4D's complex state viewed as a real vector by stacking real and imaginary parts, or Mamba's per-channel real state — total capacity is bounded above by N (Jaeger, 2001): a linear readout cannot recover more independent quantities from the state's history than the state has dimensions, so no set of lags can jointly exceed this bound.

B S4D: ADDITIONAL EXPERIMENTAL DETAIL

Full per-setting detail for the S4D summary in Section 4.1.

Table 12: Single-layer memory capacity, 5 seeds (mean ± std). Easy: $d_{\text{state}} = 64$, 32 lags. Hard: $d_{\text{state}} = 8$, $d_{\text{model}} = 8$, 128 lags.

Regime	Free capacity	Tied capacity	Free params	Tied params
Easy (ceiling)	31.997 ± 0.001	31.997 ± 0.002	24,640	16,448 (-33%)
Hard (undercapacity)	8.048 ± 0.034	8.047 ± 0.026	392	264 (-33%)

B.1 Single linear SSM layer

We compare free and tied configurations in two regimes: an *easy* regime where the state dimension exceeds the number of lags tested, and a *hard, undercapacity* regime where the number of lags far exceeds the state's capacity, producing a genuine capacity falloff curve rather than a ceiling effect.

The full per-lag capacity curve for the hard regime is reported directly in the main text as Figure 1 (Section 4.1) rather than duplicated here.

B.2 Stacked SSM with nonlinearity

A single linear layer is a best case for tying, since the readout is still a free linear combination of channels. We next ask whether the result survives once SSM layers are embedded inside a realistic nonlinear residual architecture, with $d_{\text{model}} = 32$, $d_{\text{state}} = 16$, evaluated on the same memory-capacity task at 64 lags.

Table 13: Stacked S4-style blocks (LayerNorm $\rightarrow$ depthwise SSM $\rightarrow$ GLU $\rightarrow$ residual), 5 seeds (mean ± std).

Layers	Free capacity	Tied capacity	Gap/SE	Savings
1	26.002 ± 0.652	26.241 ± 0.790	-0.52	-33%
2	32.065 ± 0.018	32.064 ± 0.021	0.03	-33%
3	32.078 ± 0.015	32.077 ± 0.007	0.06	-33%

Free and tied are statistically indistinguishable at every depth tested, including a single isolated layer (gap/SE well under 1 throughout, and the sign of the already-tiny gap even flips between depths).

B.3 Real task: sequential MNIST

Digit classification from a pixel-by-pixel scan of MNIST images (784 timesteps), using the 3-layer block stack above with $d_{\text{model}} = 128$, $d_{\text{state}} = 64$, trained for 15 epochs (Adam, batch size 128, learning rate 2×10^{-3}).

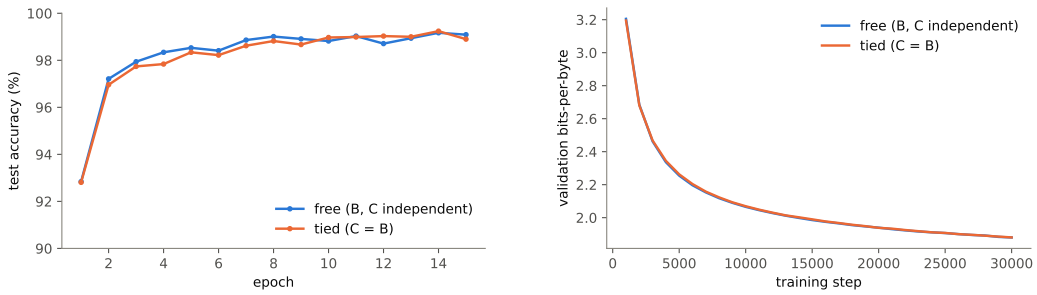
Table 14: Sequential MNIST, 3 seeds (mean ± std).

Config	Final-epoch acc.	Best-epoch acc.	SSM params	Total params
Free	$99.15\% \pm 0.10$	$99.19\% \pm 0.08$	147,840	249,482
Tied	$99.01\% \pm 0.20$	$99.19\% \pm 0.04$	98,688 (-33%)	200,330 (-19.7%)

The 0.14-point gap in final-epoch accuracy is smaller than the tied configuration's own seed-to-seed standard deviation (0.20 points), and best-epoch accuracy is identical between the two configurations to two decimal places.

B.4 Language modeling on enwik8

The architecture is $\text{Embedding}(256, d_{\text{model}}) \rightarrow n_{\text{layers}} \times \text{SSMBlock} \rightarrow \text{LayerNorm} \rightarrow \text{Linear}(d_{\text{model}}, 256)$, reusing the exact block from Appendix B.2, at $d_{\text{model}} = 256$, $d_{\text{state}} = 64$, $n_{\text{layers}} = 6$, context length $T = 1024$, batch size 32, 30k steps ($\sim$39 min/config on an A30).



(a) Sequential MNIST test accuracy per epoch (representative seed).

(b) enwik8 validation bits-per-byte over training (representative seed).

Figure 3: S4D real-task training curves: sequential MNIST (left) and enwik8 language modeling (right).

Table 15: S4D enwik8 language modeling, 3 seeds (mean ± std).

Config	Final val. bits-per-byte	SSM params
Free	1.8800 ± 0.0016	591,360
Tied	1.8824 ± 0.0016	394,752 (-33%)

Free is nominally ahead in all 3 seeds by a small, consistent margin (gap 0.0024 bpb, $\sim$0.13% relative), but gap/SE ≈ 1.5 — not statistically significant. This is a modest-scale model ($\sim$1.5M parameters, $\sim$1.88 bpb, well short of published enwik8 state-of-the-art) by design — the goal is a controlled free-vs-tied comparison, not competitive compression.

C MAMBA: ADDITIONAL EXPERIMENTAL DETAIL

Full per-setting detail for the Mamba summary in Sections 4.2–4.4.

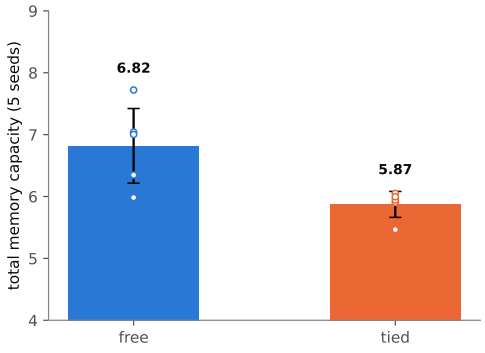
C.1 MEMORY CAPACITY: THE 25-SEED INSTABILITY EXTENSION

While diagnosing the post-hoc recovery gap (Section 5), we found that training *free* for longer than 800 steps was itself unstable (2 of 5 seeds collapsed badly), raising a real question about the main-text claim: was the clean 800-step reference just a lucky draw? We checked directly with 25 additional fresh seeds at the exact 800-step protocol: free reached 6.898 ± 1.121, with 2 of 25 seeds landing in a genuinely bad outcome ($\sim$3.1–3.7, roughly half the typical capacity); tied reached 5.892 ± 0.167 over the identical 25 seeds, with zero such failures. Pooling all 30 available seeds per config gives a *more* confident version of the original claim, not a weaker one: gap $= 0.995$ (14.5% relative, matching the original 13.9%), gap/SE $= 5.11$ (versus the original 5-seed estimate of ≈ 3.3). We do not know whether this is a known failure mode of the selective scan specifically (as opposed to this particular parallel-scan implementation or synthetic task); we have not seen it characterized elsewhere.

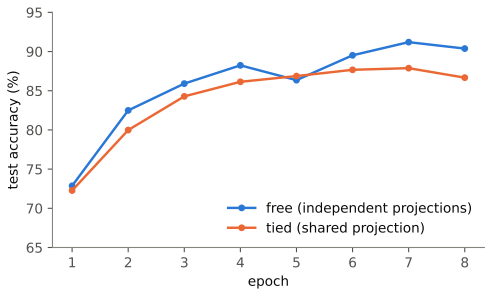
C.2 SEQUENTIAL MNIST

A single Mamba block at sMNIST scale ($d_{\text{model}} = d_{\text{inner}} = 64$, $d_{\text{state}} = 16$, 8 epochs, batch 32), 3 seeds with matched per-config reseeding:

Free wins in every seed on both metrics (6/6 head-to-head), a consistent direction, but the gap/SE ratios fall well short of the memory-capacity result's 5.11 (30 pooled seeds), and neither configuration has converged by epoch 8. This is in tension with the tied_sign-inclusive comparison reported in the main text as Table 3 (Section 4.3), run on different GPU hardware (2×A30 rather than the 4×GTX 1080 Ti used above), which instead found free and tied statistically indistinguishable (a 0.05-point



(a) Memory capacity, mean ± std over 5 seeds (outlined points: individual seeds).



(b) sMNIST test accuracy per epoch (representative seed); neither configuration has converged by epoch 8.

Figure 4: Single-block Mamba results: memory capacity (left) and sequential MNIST (right).

Table 16: Mamba sequential MNIST, single block, free-vs-tied only, 3 seeds (mean ± std), 4×GTX 1080 Ti.

Metric	Free	Tied	Gap/SE
Final-epoch test acc.	$89.73\% \pm 1.29$	$88.40\% \pm 1.23$	1.29
Best-epoch test acc.	$90.00\% \pm 1.47$	$89.05\% \pm 1.52$	0.78

final-epoch gap, an order of magnitude smaller than either config's own seed-to-seed standard deviation). This training regime is separately known to be sensitive to exact floating-point reduction order at a fixed seed, so hardware alone plausibly explains the discrepancy.

C.3 Stacking

Table 4 (Section 4.3) gives the full stacked-depth numbers. $L = 1$ reproduces the single-block study closely — free 6.908 ± 0.511 vs. that study's 6.819 ± 0.603, tied 5.872 ± 0.277 vs. 5.872 ± 0.210 — confirming this script's setup matches. The free-vs-tied gaps are 15.0%, 8.6%, and 10.3% relative at $L = 1, 2, 3$; tied_sign nominally trails free at $L = 1$ but leads at $L = 2, 3$, never crossing a significance threshold either way.

C.4 SlimPajama scaling-ladder detail

The ∼29.3M-parameter check ($d_{\text{model}} = 640$, $d_{\text{inner}} = 1280$, $d_{\text{state}} = 16$, 11 layers, single seed, 1000 steps): free reached 2.1822 bpb, tied 2.1920 bpb. The ∼150M point (Table 6, Section 4.4): $d_{\text{model}} = 1088$, $d_{\text{inner}} = 2176$, $d_{\text{state}} = 16$, $dt_{\text{rank}} = 68$, 20 layers, $T = 512$, batch 4, checkpointed. Both tied and tied_sign there are nominally ahead of free by -0.30% and -0.16% relative, the same opposite-sign pattern as the single-seed 300M check (Appendix I), now confirmed at 3 seeds across both tying conditions; all three gaps are smaller than the seed-to-seed spread.

D Vanilla RNN: full results

Full detail for the third-architecture check summarized in Section 6: a plain (Elman) RNN, $h_t = \tanh(W_{hh} h_{t-1} + W_{ih} \odot u_t)$, $y_t = \sum_n (h_t \odot W_{ho})_n$, with $W_{ih}, W_{ho} \in \mathbb{R}^{D \times N}$ playing the B/C role and W_{hh} (a dense $N \times N$ matrix per channel) untouched by the tie in either configuration; tying sets $W_{ho} = W_{ih}$ literally. The core memory-capacity numbers are now reported directly in the main text as Table 9 (Section 6) rather than duplicated here: single-layer gaps are 0.15% and 1.45% (single seed), and the stacked means match almost exactly at 1, 2, and 3 layers once averaged over 3 seeds. A side finding at 1 layer: free's seed-to-seed standard deviation on total capacity is ∼100× tied's (0.81 vs. 0.006) — tying appears to stabilize training of an isolated nonlinear recurrent layer even

though it does not move the mean — but this variance gap closes by 2 layers and has only been checked at one small scale.

On sequential MNIST (single `RNNBlock`, matching the S4-style block structure used throughout this paper, 10 seeds), both configurations plateau far below the SSM results (∼45–60% vs. ∼99%) — an expected vanilla-RNN vanishing-gradient weakness at 784 steps, not a tying artifact. At the original $n = 3$, tied trailed free by 1.6–2.2 points (final-acc $51.11\% \pm 6.07$ vs. $48.94\% \pm 10.23$), a gap smaller than either config's own seed-to-seed standard deviation. Extending to $n = 10$ reverses the nominal direction rather than confirming it: tied now leads by about a point on both metrics (final-acc $49.37\% \pm 10.83$ vs. $50.39\% \pm 8.52$, gap/SE ≈ -0.23; best-acc $52.73\% \pm 7.37$ vs. $54.09\% \pm 5.48$, gap/SE ≈ -0.47), still nowhere near significance either way. The sign flip across sample sizes is itself informative: a real effect does not typically reverse direction as more independent seeds are added, while noise does.

E Parameter-matched width reduction: full results

Full detail for the parameter-matched width-reduction check summarized in Section 4.3: all six memory-capacity rounds, the single-block (L=1) history that motivated stacking to $L = 3$ in the main text, the full four-round sMNIST comparison, and the abandoned memory-constrained language-model attempt.

E.1 Memory capacity: all six rounds, and the single-block history

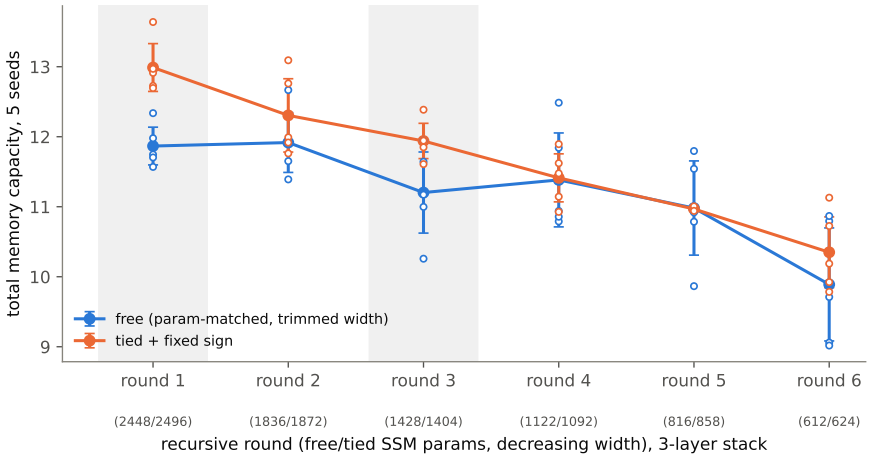


Figure 5: Six rounds of the recursive width-matched check, memory capacity, 3-layer stack, 5 seeds per condition per round. SSM parameter counts (free/tied) are annotated per round; width (d_{inner}) decreases left to right: free 24→18→14→11→8→6, tied_sign 32→24→18→14→11→8. Shaded bands: the two rounds where tied_sign is significantly better than free.

Table 17: Recursive width-matched check, memory capacity, 3-layer stack, mean ± std over 5 seeds, all six rounds. Gap/SE is free minus tied_sign.

Round	Tied_sign (params)	$\text{Free}_{\text{trimmed}}$ (params)	Gap/SE
1	12.988 ± 0.382 (2496)	11.867 ± 0.301 (2448)	-5.15
2	12.305 ± 0.585 (1872)	11.917 ± 0.477 (1836)	-1.15
3	11.940 ± 0.282 (1404)	11.203 ± 0.648 (1428)	-2.33
4	11.412 ± 0.384 (1092)	11.383 ± 0.750 (1122)	-0.08
5	10.968 ± 0.026 (858)	10.982 ± 0.752 (816)	$+0.04$
6	10.350 ± 0.565 (624)	9.890 ± 0.903 (612)	-0.97

At $L = 3$, tied_sign is never significantly beaten by the width-matched free model, at any of the six rounds tested. Rounds 1 and 3, at the widest widths, show tied_sign significantly *ahead* (gap/SE -5.15, -2.33); the remaining four rounds are statistical washes. This is a materially different picture from the single-block ($L = 1$) version of this same check: there, an identical recursive procedure at the identical width schedule found a real, significant *free-model win* in the middle of the range (gap/SE $+4.02$, $+2.80$), bounded by a wash on both the wide and narrow sides — a shallow-depth artifact that does not survive stacking to $L = 3$, mirroring S4D's own illusory single-layer tying cost (Appendix B.2).

E.2 Sequential MNIST: full four-round results

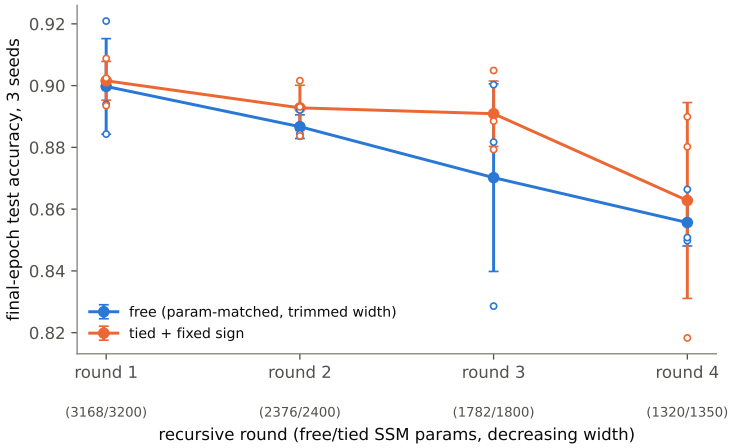


Figure 6: Four rounds of the recursive width-matched check, sMNIST, 3 seeds per condition per round. SSM parameter counts (free/tied) are annotated per round; width (d_{inner}) decreases left to right: free $48\to36\to27\to20$, tied_sign $64\to48\to36\to27$.

Table 18: Recursive width-matched check, sMNIST final-epoch accuracy, mean $\pm$ std over 3 seeds. Gap/SE is free minus tied_sign.

Round	Tied_sign (params)	$\text{Free}_{\text{trimmed}}$ (params)	Gap/SE
1	0.9016 ± 0.0077 (3200)	0.8997 ± 0.0190 (3168)	-0.16
2	0.8928 ± 0.0090 (2400)	0.8867 ± 0.0047 (2376)	-1.05
3	0.8909 ± 0.0130 (1800)	0.8702 ± 0.0372 (1782)	-0.91
4	0.8628 ± 0.0388 (1350)	0.8557 ± 0.0093 (1320)	-0.31

sMNIST shows no reversal at any round tested: tied_sign is nominally ahead of the width-matched free model at every single width, never significantly so, but never once behind. This check was run at a single block ($L = 1$), not stacked to $L = 3$ like the memory-capacity comparison above, due to cost.

E.3 Language model: an abandoned memory-constrained attempt

Before the properly-powered small-scale LM check reported in the main text, we first attempted the identical recursive procedure at the $\sim$29.3M-parameter SlimPajama configuration used elsewhere in this paper. Mamba's parallel scan is memory-hungry enough at this width that even batch size 2 exhausts an 11GB GPU, forcing batch size 1 for the whole comparison. Under that regime, round 1 (the widest width) showed a significant *free-model win* (gap/SE $+3.25$), and subsequent rounds bounced around inconsistently (gap/SE -0.37, $+1.82$) — a pattern that looked less like a real effect and more like the kind of unreliable measurement this project has repeatedly needed to catch and correct elsewhere (e.g. a `tied_sign` mask reseeding bug in the post-hoc sign-fix study, where re-running nominally the same seed silently produced a different mask and a visibly different number).

We reran the identical procedure at the smaller, properly-powered configuration reported in the main text instead; the contrast between the two attempts is itself a useful methodological finding, illustrating why this paper insists on multi-seed, adequately-batched comparisons before trusting any single result.

F QKV PROJECTION-SHARING COMPARISON

Full detail for the attention comparison summarized in Section 6 (Kayyam et al., 2026): tying $K = V$ while leaving Q independent, versus collapsing all three projections to a single shared one ($Q = K = V$), evaluated on a synthetic algorithmic-reasoning suite averaged over five tasks (REVERSE, SORT, SUBTRACT, SWAP, COPY). The accuracy numbers are now reported directly in the main text as Table 11 (Section 6) rather than duplicated here: tying two of three projections ($Q \neq K = V$) costs $< 1\%$ relative accuracy (0.751 vs. 0.758); collapsing all three ($Q = K = V$) costs substantially more (0.651, a 14% relative drop).

G POST-HOC RECOVERY: DIAGNOSTIC DETAIL

Full ablation detail for the diagnosis summarized in Section 5: why does recovered post-hoc tied initially lag free, and what fixes it? All rows below use the memory-capacity task, 5 seeds.

Table 19: Ruling out a co-adaptation mismatch by fixing it two ways, 2000 steps.

Condition	Total memory capacity
Baseline (full carryover, joint fine-tune throughout)	6.220 ± 0.399
Reset every weight except the merged projection, then fine-tune	6.555 ± 0.216
Warmup (freeze all but the projection for 400 steps, then unfreeze)	6.339 ± 0.341
Reference: tied trained from scratch, 2000 steps	6.535 ± 0.316
Reference: free	6.819 ± 0.603

Resetting the co-adapted weights closes the gap on its own (matching the trained-from-scratch reference almost exactly), confirming the diagnosis.

Table 20: Learning-rate ablation: does a bigger step close the gap, or only a longer walk?

Condition	Total memory capacity
Baseline (standard learning rate, 2000 steps)	6.220 ± 0.399
Higher learning rate ($10\times$, 2000 steps)	5.883 ± 0.516
Much higher learning rate ($30\times$, 2000 steps)	5.517 ± 0.615
Differential rate (projection standard, other weights $10\times$)	4.981 ± 0.817
Standard rate, 6000 steps instead of 2000	$\mathbf{6.742 \pm 0.333}$
Reference: free	6.819 ± 0.603

Every attempt to accelerate convergence with a larger step backfires monotonically; only patience (more steps at the same, standard rate) closes the gap, confirming the co-adaptation mismatch is a longer walk, not a wall requiring a bigger step. This is why the main text reports the 6000-step recovery number throughout, not the 2000-step one used here to isolate the mechanism.

The sign-fix recovery ranking at both budgets, showing this reversal, is reported directly in the main text as Table 8 (Section 5) rather than duplicated here.

The sign fixes are actively catastrophic zero-shot at this scale — `tied_sign_alt` reaches 9.634 bpb, worse than an untrained model's $\log_2 256 = 8$, since a mask never adapted to this specific checkpoint scrambles its output distribution completely. Despite this, recovery brings `tied_sign_alt` within a hair of plain tied's numbers either way (real data: 2.1512 vs. 2.1377; data-free: 2.2580 vs. 2.2491) — fine-tuning washes out nearly all of the zero-shot starting condi-

Table 21: Zero-shot post-hoc conversions and full recovery detail, enwik8, single seed.

Condition	Val bits-per-byte
Free	2.1378
Zero-shot tied (merge = average)	2.7157
Zero-shot tied (keep B only)	3.5675
Zero-shot tied (keep C only)	3.6358
Zero-shot `tied_sign` (random mask)	6.4942
Zero-shot `tied_sign_alt` (deterministic mask)	9.6340
Tied, recovered (real data)	2.1377
Tied, recovered (data-free)	2.2491
`tied_sign_alt`, recovered (real data)	2.1512
`tied_sign_alt`, recovered (data-free)	2.2580

tion, echoing the memory-capacity result above: the specific post-hoc conversion barely matters once real fine-tuning happens, only the recovery recipe (real data vs. data-free) and step budget do.

H Parameter accounting: how much of the model is A, B, and C?

Every result in this paper reports tying's effect on the SSM layer's *own* parameter count (a consistent one third for S4D, a config-dependent fraction for Mamba). That is not the same question as how much of a real, deployed model's *total* parameter count is affected, since every block also contains non-SSM projections. Table 10 (Section 6) answers the second question directly, with exact counts for every real-corpus language-modeling configuration used in this paper; this appendix gives the per-block formulas behind that table and a caveat on inference cost.

Per-block parameter formulas. For a Mamba block with width d_{model}, inner width d_{inner}, state size d_{state}, and rank d_{rank}: A costs $d_{\text{inner}}d_{\text{state}}$; B and C are each $d_{\text{state}}d_{\text{inner}}$ (present twice in *free* mode, once in *tied*); the step size's path costs $d_{\text{inner}}d_{\text{rank}} + d_{\text{inner}}(d_{\text{rank}} + 1)$; the skip term D costs d_{inner}. Outside the SSM itself, `in_proj` costs $2d_{\text{model}}d_{\text{inner}}$, `out_proj` costs $d_{\text{model}}d_{\text{inner}}$, the depthwise causal conv costs $5d_{\text{inner}}$, and LayerNorm costs $2d_{\text{model}}$. For an S4D-style block with width d_{model} and state size d_{state}: A, B, and C each cost $2d_{\text{model}}d_{\text{state}}$ (real and imaginary parts), $\log \Delta$ costs d_{model}, and the non-SSM `out_proj` costs $2d_{\text{model}}^2 + 2d_{\text{model}}$.

A caveat on inference cost. This total-parameter accounting should not be conflated with per-token inference cost: during autoregressive decoding, every weight matrix in the model must be read from memory once per generated token, so total parameter count is also the right proxy for weight-loading memory traffic. What stays constant regardless of tying is the recurrent *state* h_t's size ($D \times N$ per layer, independent of sequence length) — SSMs' real advantage over attention's growing KV cache — but tying $B{=}C$ changes weight count, not state size, so it does not add to that advantage.

I Mamba SlimPajama byte-level LM at 300M parameters (single seed)

Full detail for the 300M-parameter scaling point summarized in Section 4.4: $d_{\text{model}} = 1536$, $d_{\text{inner}} = 3072$, $d_{\text{state}} = 16$, $dt_{\text{rank}} = 96$, 20 layers, ∼299M parameters (free) / ∼298M (tied, tied_sign) — the same 300M parameter class as the QKV study's headline scale. Batch size 2, $T = 512$, 1000 steps, single seed per condition, ∼11.2s/step under per-block activation checkpointing.

Unlike every smaller-scale Mamba result in this paper, tied is nominally *ahead* of free here (gap -0.0200 bpb, ∼0.80% relative), with tied_sign landing in between. This result is single-seed only, for the same deliberate cost/value reason as the 29.3M point (Appendix C) — unlike that point, the enwik8 base case and the 150M point are both multiseeded: running the three conditions in parallel across three GPUs, a single round already costs ∼3.2 wall-clock hours, and a 3-seed confirmation would cost roughly 6 more hours on top — weighed against that cost, an effect this small sits

Table 22: Mamba SlimPajama byte-level LM, ∼300M parameters, single seed (deliberately not multiseeded, see discussion below).

Config	Final val. bits-per-byte	SSM params
Free	2.5042	14,868,480
Tied	2.4842	13,885,440 (−6.6%)
Tied + fixed sign	2.4905	13,885,440 (+0 vs. tied)

squarely in the range this paper has repeatedly found to be noise at LM scale once multiseeded. The sign of that small gap at any single scale point is not something a single seed can settle, and we do not read this result as evidence that tying is better than free at this scale, only that it is not reliably worse.