Learning Rate Transfer for Hybrid Transformer-SSM Architectures

Jimin Seo[1] **Gyubok Lee**[2*] **Yeonsik Jo**[3] **Kiwoong Yoo**[3] **Yeongoon Kim**[1]
Minhae Oh[1] **Jin Woo Koo**[1] **Suhwan Kim**[1] **Nakyung Lee**[1] **Minsik Seol**[1]
Idris Nechnech[1] **Jaehyeon Kim**[4] **Giho Lee**[4] **Jungwoo Lee**[1,4†]

[1]Seoul National University [2]SB Intuitions [3]LG AI Research [4]Hodoo AI

{jimin.seo, ygoonkim, minhae.oh, jinukoo, curtis05, leena, seolminsik}@cml.snu.ac.kr
{inechnech, junglee}@snu.ac.kr, gyubok.lee@sbintuitions.co.jp
{yeonsik.jo, kiwoong.yoo}@lgresearch.ai, {jaehyeon.kim, giho.lee}@hodooai.com

Abstract

We study learning rate (LR) scaling for hybrid architectures combining Transformer and State-Space Model (SSM) blocks, a class adopted by several recent production language models. In particular, we focus on the gap between the theoretical scaling rules derived for SSMs under zero-order-hold (ZOH) discretization at infinite width with growing state size, and the field-standard practical implementations using simplified-ZOH Mamba at fixed state size. Surprisingly, in this practical regime hybrid architectures achieve a **near-zero LR transfer gap** across widths 256–2048 and depths 4–32 up to billion-parameter scale using only the original μP prescription, even though SSM operations fall outside its Tensor Programs representability conditions and every parameterization we test fails the standard coordinate-check diagnostic of μP correctness. We attribute this to a two-condition decomposition of LR transfer in hybrid architectures: a *global* update-to-weight invariance, enforced by μP's initialization and LR scaling; and a *local* per-component balance, provided by AdamW's per-parameter normalization. Our observations show that the optimal LR is invariant to width up to $8\times$, that this width invariance holds across depth, sequence length, batch size, and Transformer-to-SSM ratio, and that it transfers to Nemotron-H, a production hybrid outside our custom architecture set. We hope these findings fill the gap between theoretical scaling rules and practical hybrid implementations, and stimulate further research toward bridging it.[3]

1 Introduction

Hybrid architectures combining Transformer attention with State-Space Model (SSM) blocks have emerged as a promising direction for large language models, pairing attention's quality on long-range dependencies with SSMs' subquadratic inference cost. Controlled comparisons at 8B parameters show Mamba-2-Hybrid outperforming pure Transformers across benchmarks while running up to $8\times$ faster [Waleffe et al., 2024], and similar trends appear in recent production hybrids [NVIDIA, 2025, Jamba Team, 2025, Zuo et al., 2025]; a fundamental practical problem remains, however: *training these models requires expensive per-scale hyperparameter tuning.*

The learning rate (LR) must be retuned at each model scale because the optimum shifts with architectural choices like width and depth; at production scale this sweep can substantially exceed the

*Work done prior to joining SB Intuitions.

†Corresponding author.

[3]Code is available at `https://github.com/argyroneta-aquatica/hybrid-mup`.

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

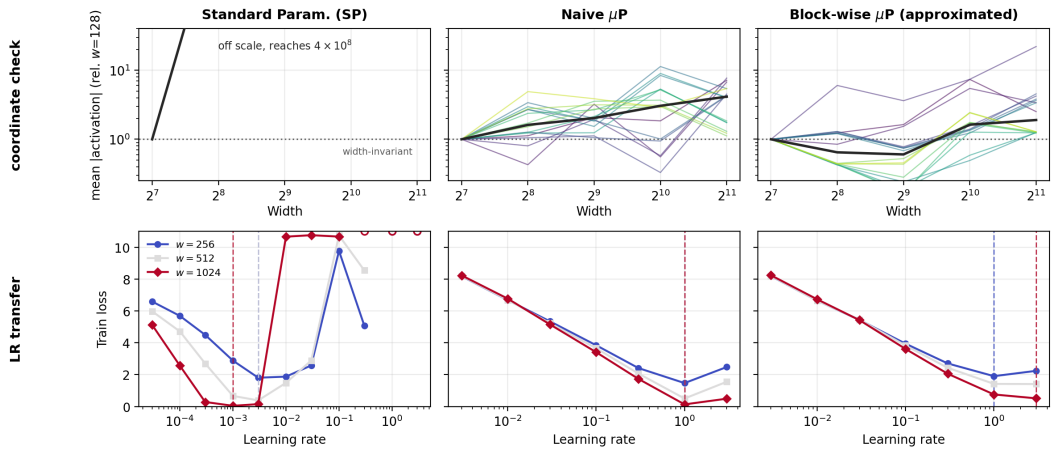


Figure 1: The coordinate check and LR transfer disagree on hybrids (MHMH, d=16). **Top:** mean absolute activation per block after 10 steps on WIKITEXT-103, relative to w=128; dotted line is width invariance, bold line the median over blocks. **Bottom:** training loss against learning rate on WIKITEXT-2 (2k steps), each width's optimum dashed and diverged runs open. The diagnostic separates SP from both μP variants, while transfer separates naïve μP, which holds η^*=1.0 at every width, from the other two.

single-run training budget [Yang et al., 2021]. Maximal Update Parameterization (μP) [Yang et al., 2021] solves this for pure Transformers by scaling initialization and LR so the optimum LR becomes width-independent. Chennuru Vankadara et al. [2024] showed that SSM operations fall outside the Tensor Programs framework and derived component-specific rules (μP-SSM), validated on pure SSMs. To our knowledge, no prior work addresses learning rate transfer for **hybrid architectures** at **practical finite widths**.

Without any SSM-specific extension, the original μP prescription (which we refer to as *naïve* μP) under AdamW [Loshchilov and Hutter, 2019] achieves **near-zero learning rate transfer gap** across all tested hybrid architectures, up to 8× width (256 to 2048) and 8× depth (4 to 32).[4] Strikingly, the standard coordinate-check diagnostic does not predict LR transfer in this regime (Figure 1): every parameterization we test fails the coordinate check on hybrids, yet naïve μP transfers cleanly while μP-SSM [Chennuru Vankadara et al., 2024] (designed for pure SSMs at infinite width under SGD with true-ZOH and growing N_x) does not. Specifically, μP-SSM's η^* drifts with width on every SSM-bearing architecture (Table 1), and at fixed η it produces a 5–8% higher convergence loss than naïve μP on WIKITEXT-103. Section 5 interprets this observation in a two-condition decomposition: naïve μP delivers a global update-to-weight invariance, and AdamW's per-parameter normalization $g/\sqrt{v}$ delivers the local per-component balance directly across the SSM boundary, while μP-SSM's explicit per-component LR rules do not deliver this balance in our regime (an optimizer ablation at a fixed parameterization supports this attribution).

Our contributions are threefold:

1. **We empirically establish** that naïve μP with AdamW achieves near-zero LR transfer gap across hybrid architectures, widths 256–2048 (8×), and depths 4–32 up to billion-parameter scale (Section 4; Appendix C.3).
2. **We propose a two-condition decomposition** of LR transfer (global update-to-weight invariance, from μP; local per-component balance, which Adam satisfies automatically), grounded by a per-component scaling test showing Vankadara's prescription works at initialization for B and C but fails for $A_{\log}$ under the field-standard simplified-ZOH Mamba implementation (Section 5, Appendix E.2).
3. **We validate generalization** to a production hybrid model [NVIDIA, 2025], which transfers at 0.0% gap without architecture-specific tuning (Section 4.3).

[4] Measured on the 7-point grid $\eta \in \{0.003, 0.01, 0.03, 0.1, 0.3, 1.0, 3.0\}$ where η^*=1 at every cell; a finer sweep (Appendix C) adds intermediate points $\{0.5, 0.7, 1.5, 2.0\}$ and shows the basin between η=0.5 and η=1.5.

2 Background and Related Work

2.1 Maximal Update Parameterization (μP)

Under standard parameterization (SP), weight matrices are initialized as $W \sim \mathcal{N}(0, 1/\sqrt{d})$ per element with a single learning rate η shared across all parameters; as fan-in width d grows, the update-to-weight ratio $\|\Delta W\|_F/\|W\|_F$ shifts with d and the optimal LR has to be re-tuned at each target scale. Yang et al. [2021] introduced Maximal Update Parameterization (μP) within the Tensor Programs framework, which scales hidden-layer initialization to $W \sim \mathcal{N}(0, 1/d)$ and learning rates to η_{base}/d so this ratio stays $\mathcal{O}(1)$ as $d \to \infty$, allowing the LR found at a small *proxy* width to transfer directly to the target. The Tensor Programs framework gives a rigorous infinite-width limit for architectures composed of *representable* operations, and it was validated for pure Transformers up to 6.7B parameters [Yang et al., 2021] and studied empirically up to 10B [Lingle, 2024]. Yang and Littwin [2023] extended the theory to Adam [Kingma and Ba, 2015] and other entrywise optimizers, characterizing μP as the unique maximal-update (feature-learning) limit for these optimizers. Subsequent works extend μP to other architectures and optimizer families [Dey et al., 2024, Zheng et al., 2025, Małaśnicki et al., 2026, Qiu et al., 2025]; Hayou [2026] gives the first formal width-transfer proof for linear MLP, and Noci et al. [2024] provide a complementary spectral analysis showing μP induces a *super consistency* of the loss-landscape sharpness across widths.

2.2 μP-SSM: Scaling Rules for State-Space Models

State space models are the family of architectures built on continuous-time linear dynamical systems [Gu et al., 2020, 2022], represented by Mamba [Gu and Dao, 2024]. Chennuru Vankadara et al. [2024] showed that three operations in the Mamba block violate the representability conditions of the Tensor Programs framework, so standard μP does not guarantee width-independent dynamics for SSMs. They derived SSM-specific scaling rules (μP-SSM) such that each component's contribution to the output remains constant with width; their analysis establishes that SSMs fail the feature learning condition under *both* SGD and Adam in the infinite-width limit, with empirical validation under SGD on pure S4 and Mamba.

Three gaps motivate extending μP to hybrid architectures. **(1) Pure-SSM only**: Chennuru Vankadara et al. [2024]'s Assumption 3.1 takes all non-SSM components to already follow μP, focusing the analysis on the SSM layer alone; the interaction between well-parameterized Transformer blocks and SSM blocks in a hybrid was not tested. **(2) Infinite-width asymptotic**: their claims hold as $N_u, N_x \to \infty$ (where N_u is the SSM operating width and N_x the state dimension), and the asymptotic failure may not manifest at the practical finite widths (256–2048) we test, particularly when AdamW's per-parameter normalization ($g/\sqrt{v}$) compensates. **(3) True-ZOH with growing N_x**: the derivation assumes the true zero-order-hold discretization with state size $N_x \to \infty$, while the field-standard Mamba implementation [Gu and Dao, 2024] uses a simplified-ZOH at fixed $N_x{=}16$ regardless of width.

2.3 Adam and Hyperparameter Transfer

Tomihari and Sato [2026] prove the formal mechanism on pure Transformers: Adam's coordinate-wise normalization makes updates depend on gradient signs, conferring robustness to cross-block heterogeneity. We show this mechanism extends across the Transformer-SSM boundary in our setting, where gradient magnitudes differ widely across SSM and Transformer components (Appendix E.2). Kosson et al. [2026] identify weight decay as the primary stabilizer for pure Transformers at 20B-token scale. In our hybrid ablation, removing weight decay changes the loss by at most 0.4% while $g/\sqrt{v}$ is essential.[5] Mlodozeniec et al. [2026] extend HP transfer to other axes (depth, batch size, duration, per-module HP); our batch-size variation gap of $\leq 1.1\%$ is consistent with this broader picture. Everett et al. [2024] show that per-layer LR scaling under SP can match μP on Transformers, indicating that multiple paths to width-invariant LR exist; we focus on the μP-based path under AdamW for hybrid architectures.

[5]The ablation spans 2k steps, which is shorter than the weight-decay timescale $1/(\eta\lambda)$ at our target widths. It therefore does not cover the longer training horizons examined by Kosson et al. [2026].

2.4 Hybrid Architectures and Production Training

Hybrid Transformer-SSM models are increasingly used in production language models [NVIDIA, 2025, Jamba Team, 2025, Dong et al., 2025, Zuo et al., 2025] and studied as competitive alternatives to pure Transformers in controlled comparisons [Waleffe et al., 2024]; production hybrid training, however, generally relies on per-scale hyperparameter tuning rather than μP-style transfer. Falcon-H1 [Zuo et al., 2025] is the closest concurrent work that tunes hybrids at scale: they jointly optimize 35 per-component multipliers (LRs, weight decay, and forward-pass activation scales) for hybrid architectures up to 34B parameters. Their stated motivation is production training stability (e.g., scaling the Δ-projection to prevent its explosion) and per-component performance tuning, not μP-style LR transfer; their work and ours therefore solve different problems, with our work isolating the minimal sufficient ingredients for the LR-transfer subproblem under AdamW. Nemotron-H [NVIDIA, 2025], which we revisit in Section 4.3, similarly relies on per-scale hyperparameter tuning despite using a different hybrid composition (Mamba-2 [Dao and Gu, 2024] SSD blocks with $\sim$8% attention layers and squared-ReLU FFN).

3 Experimental Setup

We empirically validate μP on hybrid Transformer-SSM architectures, focusing on LR transfer across widths while maintaining training stability. Production hybrid architectures use a wide range of mixing patterns without consensus on a canonical design (Section 2.4), and training dynamics depend on block composition. Therefore we first probe LR transferability at heterogeneous block interfaces, where training dynamics are most likely to shift; then across depth and width; and finally in larger training runs with downstream evaluation.

3.1 Hybrid Transformer-SSM Architectures

We study models represented by a `layer_pattern` string where each character selects a block type: **Transformer Block (T)** is from the original Transformer architecture [Vaswani et al., 2017] with grouped-query attention [Ainslie et al., 2023], rotary position embeddings [Su et al., 2024], and a SwiGLU FFN [Shazeer, 2020]. This follows the LLaMA-style design [Touvron et al., 2023], which is the most widely used Transformer architecture in practice. **Mamba Block (M)** is a selective state-space model (S6) with causal conv1d from the original Mamba architecture [Gu and Dao, 2024][6]. **Hybrid Block (H)** is a parallel fusion of Transformer and Mamba sub-blocks with additive merge. This block is an example of the parallel fusion of attention and SSM operations with minimal configuration, but still it represents the architectural family of Zuo et al. [2025], Dong et al. [2025] that contains parallel branches. Our architecture set pairs pure baselines (`MM`, `TT`) with a serial hybrid (`MTMT`), parallel hybrids at half and full H-block density (`MHMH`, `HMHM`, `HHHH`), and a hybrid-Transformer mixture (`HTHT`); the depth sweep adds the reversed orders `TMTM` and `THTH`. `MTMT` and `MHMH` place M first, consistent with Cooper et al. [2026]; the reversed orders are included for transfer coverage, not to compare orderings. We conduct additional experiments varying the serial/parallel Transformer-to-SSM block ratio to check whether different ratios affect the LR-transfer behavior; and no effect was found. See Appendix D.6, D.5 for details.

3.2 Parameterization Schemes

We compare four schemes: Standard Parameterization (SP), Naïve μP, Block-wise μP, and Block-wise μP (approximated).

Standard Parameterization (SP): weight matrices are initialized as $W \sim \mathcal{N}(0, 1/\sqrt{d_{\text{model}}})$ per element, with a single shared learning rate η across all parameters, where d_{model} is the model hidden dimension. **Naïve μP**: embeddings $\mathcal{N}(0, 1)$, hidden layers $\mathcal{N}(0, 1/d_{\text{model}})$, output head zeros; hidden-layer LR scaled by $\eta_{\text{base}}/d_{\text{model}}$. **Block-wise μP**: naïve μP on Transformer components plus Chennuru Vankadara et al. [2024]'s Table 1 SSM prescription (per-component init scales σ_B, σ_C and learning rates η_A, η_B, η_C in N_u (SSM operating width) and N_x (state dim); explicit formulas in Appendix A.3).

[6] We used the original version of Gu and Dao [2024], but an additional experiment empirically showed that it is also applicable to Mamba-2 blocks [Dao and Gu, 2024] (Appendix C.5).

We also use the **Block-wise μP (approximated)** variant, which keeps the per-component initialization and the η_C scaling of Chennuru Vankadara et al. [2024] but scales η_A and η_B with the naïve-μP exponent (Appendix A.3; motivation in Section 5.3). Both block-wise variants use the SSM discretization from the official Mamba library [Gu and Dao, 2024], which differs from the true-ZOH form underlying Chennuru Vankadara et al. [2024]'s Table 1 derivation; we discuss the resulting regime gaps in Appendix A.4.

3.3 Training Details

All experiments use AdamW [Loshchilov and Hutter, 2019] (β_1=0.9, β_2=0.95), cosine warmup, gradient clipping at 1.0, and a fixed seed. We use WIKITEXT-2/WIKITEXT-103 [Merity et al., 2017] (2.4M/120M tokens) for short-horizon and convergence runs, and FINEWEB-EDU [Penedo et al., 2024] (500M tokens) for larger-corpus validation; per-experiment hyperparameters appear in Section 4. WIKITEXT-103 carries the controlled convergence sweeps and FINEWEB-EDU tests the recipe on more realistic web-scale data; since μP's transfer is a statement at fixed data, we make no cross-dataset LR-optimality claim. Each LR sweep uses a single tokenizer: GPT-2 BPE on WIKITEXT-2 and tiktoken `cl100k_base` on WIKITEXT-103 and FINEWEB-EDU (Appendix A.2); the baseline-matched comparison of Table 2 additionally uses the GPT-NeoX tokenizer, and its Pythia and Mamba baselines share GPT-NeoX's initialization and differ in optimizer settings. Unless a sweep varies them, our LR sweeps use sequence length 128 and batch size 64 for 2k or 20k steps (16M or 164M tokens), extended to 60k steps (492M tokens) at 1.24B parameters (Appendix C.3); the 5.0B-token run (Section 4.2) and the 24.9B-token run (Appendix C.3) use the transferred LR at sequence length 2048 and are convergence runs, not re-sweeps. Robustness experiments are in Appendix D and total project compute ($\sim$5,500 GPU-hours, with the per-architecture tuning-cost saving) is in Appendix A.1.

3.4 Evaluation Metrics

Our main metric is the **learning rate transfer gap**: the relative loss difference[7] between training the target with the proxy's optimum LR versus its own optimum,

$$\text{Transfer Gap} = \frac{\left|\mathcal{L}(\eta^*(c_{\text{proxy}}); c_{\text{target}}) - \mathcal{L}(\eta^*(c_{\text{target}}); c_{\text{target}})\right|}{\left|\mathcal{L}(\eta^*(c_{\text{target}}); c_{\text{target}})\right|}, \tag{1}$$

where $\mathcal{L}(\eta; c)$ is the end-of-training loss, $\eta^*(c) = \arg\min_\eta \mathcal{L}(\eta; c)$, and c is the configuration (architecture, width, depth, etc.). If the transfer gap is 0%, it indicates the proxy's optimal LR is also optimal at the target within the tested LR resolution.

4 Results

We validate naïve μP LR transfer through stages of increasing complexity: unit-level interface checks and depth scaling in Section 4.1, width transfer across architectures and scaling to a larger dataset in Section 4.2, and generalization to a production architecture in Section 4.3. Full per-experiment details are in Appendices B, C, and C.1; this section summarizes the main findings.

4.1 Unit Interface and Depth Validation

Hybrid architectures compose heterogeneous block types, creating interfaces (e.g., Mamba→Transformer, Hybrid→Mamba) from distinct signal propagation dynamics per block. We enumerate all 9 unit interfaces constructible from T, M, and H blocks and measure training stability across widths 128–2048 for each parameterization scheme.

[7] Following Yang et al. [2021], we measure transfer on training loss, which excludes the effect of regularization; appendix results evaluated on a held-out split report validation loss, as indicated in each case. Across all sweeps on WIKITEXT-103 and FINEWEB-EDU that were checked on both training and validation loss, the two select the same η^* in 209 of 218 cells. Where they differ, the validation loss at the training-loss optimum lies within 0.6% of the validation minimum, except in the 60k-step runs of Appendix C.3 (up to 1.7%). This difference compares the optima of the two losses within a single configuration and is not a transfer gap; the validation-loss transfer gap in those runs is 0.0%. On WIKITEXT-2, which a 2k-step run traverses about seven times, validation loss prefers a lower LR than training loss (0.1 against 1.0 for naïve μP, where the grid resolves the optimum).

Table 1: Optimal LR at each (scheme, architecture, width); uniform shading within each row means LR transfer to the corresponding widths. Values from 2k-step training on WIKITEXT-2. The SP grid for MM and MTMT starts at η=0.003.

	Naïve μP			Block-wise μP (approximated)			Block-wise μP			SP		
Arch	w=256	w=512	w=1024	w=256	w=512	w=1024	w=256	w=512	w=1024	w=256	w=512	w=1024
MM	**1.0**	**1.0**	**1.0**	1.0	3.0	3.0	3.0	3.0	1.0	0.010	0.003	0.003
TT	**1.0**	**1.0**	**1.0**	1.0	1.0	1.0	1.0	1.0	1.0	0.003	0.001	0.0003
MTMT	**1.0**	**1.0**	**1.0**	1.0	1.0	1.0	1.0	3.0	1.0	0.010	0.003	0.003
MHMH	**1.0**	**1.0**	**1.0**	1.0	3.0	3.0	3.0	3.0	1.0	0.003	0.003	0.001
HMHM	**1.0**	**1.0**	**1.0**	1.0	1.0	3.0	3.0	3.0	1.0	0.003	0.003	0.001
HHHH	**1.0**	**1.0**	**1.0**	1.0	1.0	3.0	3.0	3.0	1.0	0.003	0.003	0.001
HTHT	**1.0**	**1.0**	**1.0**	1.0	1.0	1.0	3.0	1.0	1.0	0.003	0.003	0.001
Transfers?	**Yes** (0/7 shift)			Partially (4/7 shift)			Drifts (6/6 SSM-bearing shift)			No (7/7 shift)		

The stability window of naïve μP extends beyond the highest LR tested in this sweep (η=1.0) across all widths and interfaces.[8] The stability window of SP shrinks with width, dropping from 0.30–1.0 at w=128 to 0.03–0.10 at w=2048. Block-wise μP (approximated) with η_A instead set to $\eta_{\text{base}} N_u$ collapses to $\eta \leq 0.01$ at $w \geq 256$ for serial SSM interfaces (MM, MT); Section 5.3 interprets this as a redundant per-component correction stacking with Adam's $g/\sqrt{v}$. Full stability tables are in Appendix B.

Across depths 4–32 on 8 architectures (MM, TT, MTMT, MHMH, HHHH, HTHT, TMTM, THTH; spanning pure SSM/Transformer, serial hybrids, and parallel/reversed compositions) at w=512, naïve μP keeps η^*=1.0 in all 32 cells and the approximated block-wise variant in 23, moving to η^*=3.0 on MM at every depth and in five hybrid cells at $d \leq 8$; full block-wise drifts to η^*=3.0 in 25 of the 28 SSM-bearing cells (consistent with $\eta_A \propto N_u$) and destabilizes at $\eta \geq 10$ in all 28 (Appendix B.2). The approximated variant's 20k-step loss on WIKITEXT-103 stays within 0.5% of naïve at w=1024 (Appendix A.4; interpreted in Section 5.3).

4.2 Width Transfer and Scale Validation

We train 7 architectures (MM, TT, MTMT, MHMH, HMHM, HHHH, HTHT; spanning pure SSM/Transformer baselines, serial and parallel hybrids of varying H-block density) at widths 256, 512, 1024 across all four schemes (Table 1; 20k-step WIKITEXT-103 convergence in Appendix A.4).

Naïve μP is the unique scheme that transfers cleanly: η^*=1.0 at all widths for all 7 architectures, while the other three drift with width.[9] We therefore test the joint depth-width grid only under naïve μP, building on its zero transfer gap on width (Table 1) and depth (Section 4.1) individually.

From the joint depth-width experiment, naïve μP shows η=1.0 transfer from a proxy (w=256, d=8, ∼58M params for MHMH; ∼61M for HHHH) to the full target (w=1024, d=32, ∼624M params for MHMH; ∼830M for HHHH), an approximate 11× (MHMH) to 14× (HHHH) parameter increase with 0.0% gap for both architectures (Appendix C.1). We further validate on FINEWEB-EDU (500M tokens, 20k steps) to test whether LR transfer holds on a curated web-scale corpus. Across the 3 tested architectures, naïve μP achieves 0.0% transfer gap, while SP's optimum shifts 3× smaller (0.003 → 0.001) and transferring the proxy LR without retuning incurs 1.1–8.7% penalty (Appendix C.2).

Downstream evaluation. We train MHMH on FINEWEB-EDU for 38k steps (5.0B tokens, ∼0.6× Chinchilla-optimal [Hoffmann et al., 2022]) using η=1.0 transferred from the proxy (w=256, d=16), reporting MHMH-415M (tiktoken cl100k_base) and MHMH-312M (GPT-NeoX, matching Pythia [Biderman et al., 2023]); we retrain Pythia-410M and Mamba-370M [Gu and Dao, 2024] on the same data for comparison. The goal is not to claim architectural superiority but to show that **the transferred LR produces a competitive model without retuning** (Table 2). Widths up to w=2048 are also LR-transfer validated at the 2k-step horizon (Appendix C.3). Following the standard zero-shot suite for sub-billion-parameter language models [Biderman et al., 2023, Gu and Dao, 2024], we evaluate on HellaSwag, PIQA, WinoGrande, ARC, and LAMBADA [Zellers et al., 2019, Bisk et al., 2020, Sakaguchi et al., 2020, Clark et al., 2018, Paperno et al., 2016].

[8] Extended sweeps (Table 34) confirm naïve μP trains stably at η=3 on MHMH and HHHH, and at η=10 on MHMH.

[9] The discretization used in the official Mamba library [Gu and Dao, 2024] differs from the ZOH or Euler form underlying Chennuru Vankadara et al. [2024]'s Table 1 derivation; see Section 3.2.

Table 2: Zero-shot downstream evaluation. Top: published baselines (300B tokens, The Pile). Bottom: all models retrained on FINEWEB-EDU 5B tokens. `MHMH` variants: † `MHMH`-312M uses the GPT-NeoX tokenizer (50304 vocab, tokenizer-matched to Pythia); ‡ `MHMH`-415M uses tiktoken `cl100k_base` (100277 vocab).

Model	Params	HellaSwag acc_n	PIQA acc	WinoGr acc	ARC-E acc	ARC-C acc_n	LAMBADA acc	Avg
Published baselines (The Pile, 300B tokens)								
Mamba-370M	370M	46.5	69.5	55.3	55.1	28.0	55.6	51.7
Pythia-410M	410M	40.6	66.9	53.8	52.1	24.6	51.4	48.2
Retrained on FineWeb-Edu (5B tokens)								
Mamba-370M	370M	34.6	61.6	**53.8**	55.0	26.2	19.9	41.9
Pythia-410M	410M	33.1	63.2	50.4	54.3	26.3	27.2	42.4
`MHMH`-**312M**†	**312M**	**35.1**	64.1	52.8	**57.2**	26.1	**29.2**	**44.1**
`MHMH`-**415M**‡	**415M**	34.3	**64.3**	51.9	55.9	**26.5**	27.7	43.4

Table 3: Nemotron-H width transfer (WIKITEXT-103, 20k steps, warmup=1000; $\eta \in \{1, 3, 5\}$ for naïve μP, $\{0.001, 0.003, 0.01\}$ for SP). Naïve μP achieves η^*=1.0 at all widths (matching our other architectures); SP optimum shifts from η=0.003 at w=256 to η=0.001 at w=1024.

Depth	Scheme	Best LR	w=256	w=512	w=1024	Gap
16	Naïve μP	**1.0**	3.760	3.344	**3.369**	**0.0%**
16	SP	0.001–0.003	3.775	3.382	3.356	1.9%
32	Naïve μP	**1.0**	3.527	3.313	**3.138**	**0.0%**
32	SP	0.001–0.003	3.542	3.350	3.152	1.9%

4.3 Generalization to Nemotron-H

To test whether naïve μP generalizes beyond our simple custom architectures, we validate on Nemotron-H [NVIDIA, 2025], a production hybrid using Mamba-2 [Dao and Gu, 2024], squared-ReLU FFN, and grouped-query attention [Ainslie et al., 2023] without rotary embeddings [Su et al., 2024]. We follow the Nemotron-H architecture: Mamba-2 (N), squared-ReLU FFN (F), and attention-only blocks (A) with GQA at ratio 4 and no RoPE (Appendix C.6). This architecture is designed with different block types than our set.

Naïve μP achieves η^*=1.0 across all (depth, width) combinations on Nemotron-H, matching the optimum found across our custom architectures. The transfer gap is 0.0% at every cell, within a broad basin ($\eta \in \{1, 3, 5\}$ all within $\sim 3\%$ of optimum) that makes transfer robust to LR misspecification. The optimum also stays at η=1.0 from width 512 to 2048 (980M parameters) when the MLP ratio, SSM state, Mamba-2 group count and attention head dimension are those of the released Nemotron-H-8B configuration, and two runs at about 20 tokens per parameter (57M and 299M parameters) train without divergence (Appendix C.6). SP's optimum shifts $3\times$ across width ($0.003 \to 0.001$), incurring a 1.9% transfer penalty at both depths if the proxy LR is reused.

5 Analysis

5.1 Naïve μP Violates Coordinate Checks but Learning Rate Transfers

μP's LR transfer is theoretically expected to hold only when the coordinate check is satisfied [Yang et al., 2021], i.e., when pre-activation magnitudes remain width-invariant. However, on hybrid architectures, all four parameterizations we test fail the coordinate check (Figure 8): activations vary with width across SP, naïve μP, and both block-wise variants, more than in a pure Transformer (Table 32). Despite this universal coordinate check failure, naïve μP achieves 0.0% transfer gap across every architecture in Table 1, while block-wise μP's optimum LR drifts with width on every SSM-bearing architecture. Coordinate checks therefore do not predict LR transfer on hybrids; Section 5.2 instead proposes a two-condition decomposition as an explanatory diagnostic of where transfer holds and where it breaks.

5.2 Two Conditions for Scalable Transfer in Hybrid Architectures

We propose a two-condition decomposition of LR transfer, drawing on existing theory and supported by an optimizer ablation.

Condition 1: Global update-to-weight ratio invariance. For a parameter matrix W, define $\rho(W) = \frac{\|\Delta W\|_F}{\|W\|_F}$; update-to-weight ratio invariance requires $\rho(W) = \mathcal{O}(1)$ as $d_{\text{model}} \to \infty$ for every parameter group. This is the design principle of μP: Yang et al. [2021], Yang and Littwin [2023] establish width-independent ρ, for SGD and Adam-style optimizers. Under SP, $\rho = \mathcal{O}(\eta\sqrt{d})$ grows with width, causing the optimal learning rate to shift with model width; under naïve μP, $\rho = \mathcal{O}(\eta)$ depends on the learning rate but is width-independent. Section 5.1 showed that LR transfer holds in our hybrid setting despite a width-invariance violation at the activation level, motivating the second condition below.

Condition 2: Local per-component update balance. For any pair of parameter groups W_X, W_Y at configuration c, let $\beta_{X/Y}(\eta; c) = \|\Delta W_X\|_F / \|\Delta W_Y\|_F$ denote the ratio of their per-step update magnitudes; a width-shift in β means the optimization rebalances across components as the model scales, breaking the proxy's local optimality at the target. Condition 2 requires that the proxy's optimum LR η^* preserves this ratio at the target, with deviation

$$\Delta\beta_{X/Y}(c_{\text{proxy}} \to c_{\text{target}}) = \frac{\left|\beta_{X/Y}(\eta^*; c_{\text{target}}) - \beta_{X/Y}(\eta^*; c_{\text{proxy}})\right|}{\left|\beta_{X/Y}(\eta^*; c_{\text{proxy}})\right|}, \tag{2}$$

the update-balance analogue of Equation 1: Condition 2 holds when the per-component update ratio $\beta_{X/Y}$ is preserved between proxy and target widths.

5.3 Optimizer Choice Under the Two Conditions

For Condition 1, naïve μP delivers $\rho = \mathcal{O}(\eta)$ across widths under AdamW [Yang et al., 2021, Yang and Littwin, 2023], so we treat Condition 1 as held under AdamW throughout this subsection. For Condition 2, in contrast, the choice of optimizer determines which of two normalization mechanisms is active: per-component LR scaling, or per-parameter normalization.

Under SGD: per-component LR scaling. Under SGD, $\beta_{M/T} \propto (\eta_M/\eta_T)\,\gamma_{M/T}$, where $\gamma_{M/T} = \mathrm{RMS}(\nabla_{W_M}\mathcal{L})/\mathrm{RMS}(\nabla_{W_T}\mathcal{L})$ is the gradient-RMS ratio for a Mamba/Transformer parameter pair. Width-dependence in γ must therefore be canceled by per-component LR scaling, which is the role of Chennuru Vankadara et al. [2024]'s SSM rules in their derivation regime.

Under AdamW: per-parameter normalization. Under AdamW, $\eta_{\text{eff}}(W) = \eta/\sqrt{v_W}$ normalizes each parameter's gradient by its own running RMS magnitude, so the update direction is dominated by gradient *sign* rather than magnitude (Tomihari and Sato, 2026, Thm. 4.7–4.9; Balles and Hennig, 2018, Zhang et al., 2024). The ratio $\beta_{M/T}$ then decouples from γ, and naïve μP achieves $\Delta\beta \to 0$ without any per-component LR scaling (interpretive across the SSM boundary). Table 4 confirms this and exposes a stacking failure: the (SGD, naïve μP) row gives $\Delta\beta_{A_{\log}/\text{FFN}}$=235–342%; AdamW closes the same pair to 20–21% (naïve μP row); adding Vankadara's per-component LR scaling on top inflates it to 166–436% (block-wise μP row), as the explicit $\eta_A \propto N_u$ correction stacks with the normalization AdamW already supplies.

The block-wise with SGD anomaly and its diagnosis. Symmetrically, applying Vankadara's prescription under SGD, the configuration the prescription was designed for, also produces $\Delta\beta$ above 100% on the $A_{\log}$/FFN pair (Table 4, last row). We diagnose this at the parameter granularity Vankadara's derivation assumes (B and C as separate parameter groups; Appendix E.2, Table 6). At step 1 under SGD, the faithful prescription delivers the predicted width-invariant effective update for B ($\alpha(N_u)$=+0.006) and C (α=+0.029): η_B and η_C act as designed. For $A_{\log}$, however, the faithful update has α=+1.00: the $\eta_A \propto N_u$ rule does not cancel an underlying $1/N_u$ gradient decay (no such decay is present in our gradients) but instead *amplifies* the existing N_u-dependence already at step 1. This reflects the regime gap with Vankadara's derivation: we use SGD and their per-component granularity, but use the field-standard simplified ZOH at fixed N_x=16 on a hybrid (Appendix A.4).

Table 4: Condition 2 gap $\Delta\beta_{M/T}(c_{\text{proxy}} \to c_{\text{target}})$ in %, with proxy width 128. The $A_{\log}$/FFN pair is the most width-divergent under Chennuru Vankadara et al. [2024]'s $\eta_A \in \Theta(N_u)$ rule; in_proj/FFN is a near-isotropic Mamba projection. MHMH d=16, ssm_expand=2, N_x=16, η_{base}=1.0; mean over steps 50–200 and 3 seeds. Same-seed reruns of the AdamW rows leave a run-to-run floor of about 20%; in those rows, the naïve entries and the in_proj/FFN entries lie within the floor, and the block-wise $A_{\log}$/FFN entries lie well above it.

		$A_{\log}$/FFN		in_proj/FFN	
Optimizer	Scheme	128→256	128→512	128→256	128→512
AdamW	Naïve μP	20%	21%	4%	5%
AdamW	Block-wise μP	166%	436%	15%	20%
SGD	Naïve μP	235%	342%	0%	0%
SGD	Block-wise μP	233%	835%	23%	31%

Table 5: Optimizer ablation at a fixed parameterization (HHHH at d=16, naïve-μP parameter groups for every optimizer, 2k steps on WIKITEXT-103, $\eta \in \{0.001, \ldots, 1.0\}$; per-LR sweeps in Appendix E.3). Every best loss falls at η=1.0, the top of this grid. The last column counts the LRs at which w=1024 reaches a lower loss than w=256.

Optimizer	Adaptive LR	Momentum	Best loss w=256	w=1024	w=1024 lower
AdamW	Yes	Yes (β_1=0.9)	4.392	4.099	7/7
Adam no-momentum	Yes	No (β_1=0)	4.470	4.465	7/7
SGD + momentum	No	Yes (0.9)	7.732	7.955	1/7
SGD no-momentum	No	No	9.398	10.803	2/7

Even where the prescription does work at init (for B and C), the invariance is destroyed during training: $\Delta\alpha$ from step 1 to the stable window reaches $+2.4$ ($A_{\log}$), $+1.34$ (B), $+0.68$ (C), all larger than naïve's drift ($\Delta\alpha \leq +0.7$).

AdamW supplies Condition 2 in our regime; Vankadara's rules do not. The diagnosis above identifies the ZOH regime gap as the reason Vankadara's prescription does not deliver Condition 2: under SGD, $\eta_A \propto N_u$ does not cancel its target gradient decay (no such decay is present in our gradients) and the init-level B/C invariance does not survive training; under AdamW, the rules stack on top of the normalization $g/\sqrt{v}$ already supplies, producing the over-correction in Table 4. This motivates our approximated variant (Section 3, Appendix A.3), which keeps the μP-SSM initialization but drops the $\eta_A \in \Theta(N_u)$ width-amplification. Under SGD with naïve μP, the best LR sits at the top of our 20k-step grid (η=100, Table 34) at both widths, so the grid cannot establish transfer. Condition 2 can flag such a sweep: $\Delta\beta_{A_{\log}/\text{FFN}}$ is 235–342% under SGD against 20–21% under AdamW (Table 4). An extended HHHH grid (Appendix E.3.2) places the optimum above η=100 at both widths, and η=3000 trains w=1024 but diverges at w=256.

5.4 Summary

The decomposition isolates two distinct failure modes of LR transfer in hybrid architectures: Condition 1 violations shift the optimum LR with width (the canonical μP failure mode); Condition 2 violations rebalance components across widths (a finite-width hybrid-specific failure mode). In our regime (simplified-ZOH Mamba at fixed N_x=16), the μP-SSM scaling rules do not show evidence of delivering Condition 2 under either optimizer; under SGD the rule's intended cancellation does not apply, and under AdamW the rules add scaling on top of the normalization $g/\sqrt{v}$ already supplies. AdamW's $g/\sqrt{v}$ alone delivers Condition 2 across the SSM boundary. Whether either picture changes at infinite width or under true-ZOH SSMs with growing N_x remains open.

In this section, we extracted two conditions from existing theory, and neither covers our hybrid setting under our regime. We use the two conditions as explanatory diagnostics and support this empirically (Tables 4–6, Section 5.1; Appendices E.1–E.3).

Table 6: Effective-update N_u scaling at step 1 vs the stable window (steps 100–199) under SGD, base LR $\eta = 1$. Vankadara's framework predicts $\alpha(N_u) \approx 0$ at initialization for the faithful prescription (width-invariant update). The faithful values for B and C at step 1 are bolded; $A_{\log}$ is bolded to flag the *predicted-but-not-observed* cancellation.

Component	Scheme	$\alpha(N_u)$ step 1	$\alpha(N_u)$ stable	$\Delta\alpha$
$A_{\log}$	naïve	$+0.86$	$+1.55$	$+0.68$
	faithful	$\mathbf{+1.00}$	$+3.42$	$+2.42$
B	naïve	$+0.99$	$+1.33$	$+0.34$
	faithful	$\mathbf{+0.006}$	$+1.35$	$+1.34$
C	naïve	$+1.00$	$+1.34$	$+0.33$
	faithful	$\mathbf{+0.029}$	$+0.71$	$+0.68$

6 Conclusion

We demonstrate that naïve μP with AdamW yields LR transfer across widths (up to billion-parameter scale; Appendix C.3) for hybrid Transformer-SSM architectures with near-zero transfer gap in our experimental setting, and propose a two-condition decomposition of the joint action of μP's update-to-weight invariance and the optimizer's per-component normalization. Transfer is robust to batch size, architecture pattern, training duration, sequence length, and T:M ratio, each varied individually, and generalizes to Nemotron-H, an independently designed production hybrid.

Limitations. Our results characterize a specific operating regime; three scope boundaries are worth flagging. First, billion scale, the full training horizon, and multiple seeds are validated individually rather than jointly: the LR-transfer sweep reaches 1.24B parameters (three seeds, 20k steps), and one 1.24B model trains cleanly to a Chinchilla-optimal horizon (24.9B tokens) at the transferred LR with a single LR and seed (Appendix C.3); whether the near-zero gap persists when all three hold at once remains open. Larger architectures, including Mixture-of-Experts, remain untested, and Mamba-3 [Lahoti et al., 2026] is left to future work: its complex-valued state lies outside the real-valued μP rules, and we know of no complex-state μP formulation; a hybrid built on a real-valued non-Mamba recurrent mixer, Gated Linear Attention (GLA) [Yang et al., 2024], does transfer (Appendix C.5.3). Second, we additionally ran Sophia [Liu et al., 2024], which also normalizes per element and likewise transfers (0.00% gap on `HHHH` and `MHMH` at 20k steps, three seeds), whereas Muon [Jordan et al., 2024], under its own recommended scaling, does not (Appendix E.3.7); the two differ in prescription as well as normalizer, so this does not isolate per-element normalization. Finally, two formal-theory questions remain open: whether the two-condition decomposition is jointly sufficient for transfer at infinite width, and a faithful test of μP-SSM under ZOH SSMs with growing N_x, the regime in which Chennuru Vankadara et al. [2024]'s prescription was derived (our comparisons apply it to the field-standard simplified-ZOH Mamba [Gu and Dao, 2024] at fixed $N_x = 16$; Appendix A.4).

Acknowledgments and Disclosure of Funding

This work is in part supported by the National Research Foundation of Korea (NRF, RS-2024-00451435(20%), RS-2024-00413957(20%)), Institute of Information & communications Technology Planning & Evaluation (IITP, RS-2025-02305453(15%), RS-2025-02273157(15%), RS-2025-25442149(15%), RS-2021-II211343(15%)) grant funded by the Ministry of Science and ICT (MSIT), Institute of New Media and Communications (INMAC), and the BK21 FOUR program of the Education, Artificial Intelligence Graduate School Program (Seoul National University), and Research Program for Future ICT Pioneers, Seoul National University in 2026. The authors declare no competing interests.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query Transformer models from multi-head checkpoints. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4895–4901, 2023.

Lukas Balles and Philipp Hennig. Dissecting Adam: The sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 404–413, 2018.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning (ICML)*, volume 202 of *PMLR*, 2023.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 7432–7439, 2020.

Leena Chennuru Vankadara, Jin Xu, Moritz Haas, and Volkan Cevher. On feature learning in structured state space models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 86145–86179, 2024.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

John Cooper, Ilias Diakonikolas, Mingchen Ma, and Frederic Sala. Expressivity-efficiency tradeoffs for hybrid sequence models. In *International Conference on Machine Learning (ICML)*, volume 306 of *Proceedings of Machine Learning Research*, pages 21374–21398, 2026.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 10041–10071, 2024.

Nolan Dey, Shane Bergsma, and Joel Hestness. Sparse maximal update parameterization: A holistic approach to sparse training dynamics. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 33836–33862, 2024.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan (Celine) Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A hybrid-head architecture for small language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Katie E Everett, Lechao Xiao, Mitchell Wortsman, Alexander A Alemi, Roman Novak, Peter J Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers. In *International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 12666–12700, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling (COLM)*, 2024.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent memory with optimal polynomial projections. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1474–1487, 2020.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations (ICLR)*, 2022.

Moritz Haas, Jin Xu, Volkan Cevher, and Leena Chennuru Vankadara. $\boldsymbol{\mu\mathbf{P}^2}$: Effective sharpness aware minimization requires layerwise perturbation scaling. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 38888–38959, 2024.

Moritz Haas, Sebastian Bordt, Ulrike von Luxburg, and Leena Chennuru Vankadara. On the surprising effectiveness of large learning rates under standard width scaling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Soufiane Hayou. A proof of learning rate transfer under μP. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 300 of *Proceedings of Machine Learning Research*, pages 3322–3330, 2026.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 30016–30030, 2022.

Jamba Team. Jamba: Hybrid Transformer-Mamba language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Atli Kosson, Jeremy Welborn, Yang Liu, Martin Jaggi, and Xi Chen. Weight decay may matter more than μP for learning rate transfer in practice. In *International Conference on Learning Representations (ICLR)*, volume 2026, pages 67012–67048, 2026.

Aakash Sunil Lahoti, Kevin Li, Berlin Chen, Caitlin Wang, Aviv Bick, Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *International Conference on Learning Representations (ICLR)*, volume 2026, pages 86173–86202, 2026.

Lucas Lingle. An empirical study of μP learning rate transfer. *arXiv preprint arXiv:2404.05728*, 2024.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *International Conference on Learning Representations (ICLR)*, volume 2024, pages 1621–1650, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Jan Małaśnicki, Kamil Ciebiera, Mateusz Boruń, Maciej Pióro, Jan Ludziejewski, Maciej Stefaniak, Michał Krutul, Sebastian Jaszczur, Marek Cygan, Kamil Adamczewski, and Jakub Krajewski. μ-parametrization for mixture-of-experts. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, Lecture Notes in Computer Science, pages 359–373, 2026.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Bruno Mlodozeniec, Pierre Ablin, Louis Béthune, Dan Busbridge, Michal Klein, Jason Ramapuram, and Marco Cuturi. Completed hyperparameter transfer across modules, width, depth, batch & duration. In *International Conference on Learning Representations (ICLR)*, 2026.

Lorenzo Noci, Alexandru Meterez, Thomas Hofmann, and Antonio Orvieto. Super consistency of neural network landscapes and learning rate transfer. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 102696–102743, 2024.

NVIDIA. Nemotron-H: A family of accurate and efficient hybrid Mamba-Transformer models. *arXiv preprint arXiv:2504.03624*, 2025.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1525–1534, 2016.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 30811–30849, 2024.

Shikai Qiu, Charlie Chen, Hoang Phan, Qi Lei, and Andrew Wilson. Hyperparameter transfer enables consistent gains of matrix-preconditioned optimizers across scales. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, Main Conference, pages 130867–130911, 2025.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 8732–8740, 2020.

Noam Shazeer. GLU variants improve Transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Akiyoshi Tomihari and Issei Sato. Gradient heterogeneity complements Hessian heterogeneity in Transformer optimization. *Transactions on Machine Learning Research (TMLR)*, 2026.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, Garvit Kulshreshtha, Vartika Singh, Jared Casper, Jan Kautz, Mohammad Shoeybi, and Bryan Catanzaro. An empirical study of Mamba-based language models. *arXiv preprint arXiv:2406.07887*, 2024.

Greg Yang and Etai Littwin. Tensor Programs IVb: Adaptive optimization in the infinite-width limit. *arXiv preprint arXiv:2308.01814*, 2023.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor Programs V: Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention Transformers with hardware-efficient training. In *International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 56501–56523, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4791–4800, 2019.

Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhi-Quan Luo. Why Transformers need Adam: A Hessian perspective. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 131786–131823, 2024.

Chenyu Zheng, Xinyu Zhang, Rongzhen Wang, Wei Huang, Zhi Tian, Weilin Huang, Jun Zhu, and Chongxuan Li. Scaling diffusion Transformers efficiently via μP. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Jingwei Zuo, Maksim Velikanov, Ilyas Chahed, Younes Belkada, Dhia Eddine Rhayem, Guillaume Kunsch, Hakim Hacid, Hamza Yous, Brahim Farhat, Ibrahim Khadraoui, et al. Falcon-H1: A family of hybrid-head language models redefining efficiency and performance. *arXiv preprint arXiv:2507.22448*, 2025.

Experiment Inventory

The appendix collects all per-experiment details. Table 7 indexes each experiment to its section, reports the design size (number of configurations), and the headline finding. Total compute and per-config wall times are reported in Section A.1.

Table 7: Experiment inventory across the appendix. "#" is the number of configurations in the design; "naïve / approx. / faithful" denote naïve μP, approximated block-wise μP, and the literal Vankadara prescription, respectively.

§	Experiment	#	Key finding
A.4	Faithful μP-SSM: pure-Mamba SGD + hybrid AdamW	33	Lower loss in Vankadara regime; 5–8% higher loss on hybrids
B.1	Max stable LR across 9 unit interfaces $\times$ widths $\times$ schemes	195	Naïve μP stable to η=1 on all; SP shrinks with width
B.2	Depth 4–32 across 8 architectures (w=512)	128	Naïve μP keeps η^*=1 at all depths
C.1	Joint depth$\times$width grid (proxy $\rightarrow$ target)	54	η=1 transfers d=8 w=256 $\rightarrow$ d=32 w=1024
C.2	FineWeb-Edu width transfer (500M tokens)	132	0.0% gap; SP 3$\times$ shift
C.3	Width 2048 (8$\times$ scaling; `MHMH` 1.24B, `HHHH` 1.66B params)	20	0.0% gap at 8$\times$
C.5.3	GLA–Transformer hybrid (`GTGT`) width transfer	9	η^*=1 at every width
C.3	1.24B: three seeds at 20k/60k steps, a 24.9B-token run, depth to 1.88B	53	η^*=1 at every seed; converges at 24.9B tokens
C.4	Fine-grained LR ($\eta \in \{0.5, 0.7, 1.5, 2.0\}$)	24	Curves superimpose; basin $\eta \in [0.5, 1.5]$
C.6	Nemotron-H: MHA vs GQA, released-configuration dimensions, two runs at $\sim$20 tokens per parameter	77	η^* width-independent in each variant; no divergence
D.1	Convergence at 20k steps (WIKITEXT-103)	156	0.0% gap holds; SP divergence rate 100%
D.2	Multi-seed validation (3 seeds)	36	0.0% at every seed; CV $\leq$ 1.7%
D.3	Batch size $\{8, 16, 32, 64\}$	112	Gap $\leq$1.1% across batch sizes
D.4	Sequence length $\{128, 256, 512, 1024, 2048\}$	90	η^*=1 at every seq length
D.5	Parallel T:M ratio (in-block imbalance)	252	0.0% gap, all 6 ratios
D.6	Serial T:M sweep (incl. Jamba layout)	504	Naïve μP 0.0%; SP and faithful drift
D.7	Gradient clipping $\{\text{off}, 0.5, 1.0, 5.0\}$	24	Optimal LR unchanged
E.1	Coordinate check (4 schemes $\times$ 5 widths $\times$ 5 seeds)	100	All four parameterizations fail coord check on hybrids; transfer not predicted by it
E.2	Gradient + step-1 effective update test	108	γ width-dep; faithful $\alpha(N_u)$=+1.00 for $A_{\log}$
E.3	Optimizer ablations (SGD vs AdamW, momentum, WD, ϵ; Sophia, Muon)	415	Only $g/\sqrt{v}$ is load-bearing

A Setup and Parameterization Details

A.1 Compute Budget and Tuning-Cost Savings

Total project compute: $\sim$5,500 GPU-hours across 4,500+ completed configurations on NVIDIA A6000 (48 GB), A5000 (24 GB), and A100 (80 GB) on an internal SLURM cluster, counting every job of the project from its Slurm accounting, including preliminary, failed, and rerun jobs; runs on standalone workstations are not counted.

Per-config wall time. We estimate single-config wall time on a single A6000 from Slurm accounting (HHHH at d=16, batch = 64, seq_len = 128); other architectures and GPUs are within $\pm$20% of these values. Table 8 reports the per-config means we use to convert sweep budgets to GPU-hours.

Table 8: Estimated mean wall time per config on a single A6000 (HHHH, d=16). Times rise superlinearly because the SSM scan and attention costs are non-quadratic in width; the proxy-to-target ratio is $5.5\times$ at 2k steps and $\sim 12\times$ at 20k.

Width	Params (HHHH)	2k steps / 20k steps
w=256	71M	3.1 min / $\sim$16 min
w=512	181M	7.1 min / $\sim$55 min
w=1024	518M	17.1 min / $\sim$191 min

Per-architecture LR tuning cost. The headline practical implication of naïve μP + AdamW is that practitioners can replace a per-width LR sweep at the target with the same sweep at a small proxy plus one target-width verification run. Using the wall times above, on a target width of w=1024:

- *SP baseline* (no transfer, sweep 7 LRs at w=1024): 7×17.1 min $\approx \mathbf{2.0}$ GPU-hours per architecture at 2k steps; 7×191 min $\approx \mathbf{22.3}$ GPU-hours at 20k steps.
- *Naïve μP transfer* (the same 7-LR sweep at proxy w=256, 1 verification run at w=1024): $7\times 3.1 + 1 \times 17.1 \approx 38.8$ min $\approx \mathbf{0.65}$ GPU-hours at 2k; $7\times 16 + 1 \times 191 \approx 303$ min $\approx \mathbf{5.1}$ GPU-hours at 20k.

Per-architecture saving: $\mathbf{68}\%$ reduction at the 2k-step transfer horizon ($\mathbf{3.1}\times$ speedup); $\mathbf{77}\%$ at convergence ($\mathbf{4.4}\times$). The savings grow with target width because the proxy cost stays constant while the target cost scales superlinearly: a width-2048 extrapolation using the same recipe pushes the saving to $\sim 82\%$ ($5.7\times$).

A.2 Dataset Sizes and Training Coverage

We use three text corpora across the experiments. Table 9 reports the total training-split token counts, the tokenizer used, and how many optimizer steps make one epoch under our canonical configuration (seq_len=128, batch_size=64, so $128 \times 64 = 8{,}192$ tokens per step).

Table 9: Dataset sizes and training coverage at the canonical seq_len=128, batch_size=64 setup (8,192 tokens/step). The right two columns express the step budget as a fraction (or multiple) of one full epoch.

Dataset	Train tokens	Tokenizer (vocab)	Steps/epoch	2k steps	20k steps
WIKITEXT-2	2.39M	GPT-2 BPE (50,257)	292	6.9 epochs	68 epochs
WIKITEXT-103	$\sim$120M	tiktoken cl100k_base (100,277)	14,648	14% epoch	1.37 epochs
FINEWEB-EDU 500M	500M	tiktoken cl100k_base (100,277)	61,035	3.3% epoch	33% epoch

Implications for the headline claims. The width-transfer grid (2k steps on WIKITEXT-2; main-text Table 1) covers about 7 passes through that small corpus, so the 0.0% transfer gap there is measured under data-replay rather than a fresh-data regime. The convergence-validation runs (20k steps on WIKITEXT-103; Section A.4, Section D.1) cover one full epoch with a small overflow. The FINEWEB-EDU sweeps (Section C.2) at 20k steps consume $\sim 1/3$ of the 500M-token corpus, well within a single epoch. For the 5B-token run (MHMH-415M on FINEWEB-EDU, 5B tokens, 38k steps), the canonical setup uses a larger effective batch; under the same 8,192-token-per-step accounting, 5B tokens would correspond to $\sim$610k steps.

A.3 Approximated Block-wise μP: Formula

Block-wise μP [Chennuru Vankadara et al., 2024] prescribes $\eta_A \in \Theta(N_u)$, $\eta_B \in \Theta(1/\sqrt{N_u})$, and $\eta_C \in \Theta(1/(N_x\sqrt{N_u}))$ for the SSM components $(A_{\log}, B, C)$. These asymptotic scalings must be anchored at a reference width where η_{base} is tuned; an anchored implementation at $N_u^{\text{ref}} = 512$ is reported in Appendix A.4. Under AdamW, $g/\sqrt{v}$ already supplies the per-component normalization that the $\Theta(N_u)$ amplification of η_A would provide under SGD (Section 5.2), so applying both produces double normalization that destabilizes training as N_u grows.

The approximated block-wise μP used in our experiments keeps the per-component initialization rules ($\sigma_B \in \Theta(\sqrt{N_x/N_u})$, $\sigma_C \in \Theta(1/\sqrt{N_x N_u})$) and the η_C scaling of Chennuru Vankadara et al. [2024], and scales η_A and η_B with the naïve-μP exponent:

$$\eta_{A_{\log}} = \eta_{\text{base}}/N_u, \qquad \eta_B = \eta_{\text{base}}\sqrt{N_x}/N_u, \qquad \eta_C = \eta_{\text{base}}/(N_x\sqrt{N_u}), \tag{3}$$

where $N_u = d_{\text{inner}} = d_{\text{model}} \cdot \texttt{ssm_expand}$ is the SSM inner dimension. This drops the $\Theta(N_u)$ amplification of η_A (the term redundant with AdamW's $g/\sqrt{v}$) and replaces η_B's $\Theta(1/\sqrt{N_u})$ with $\Theta(1/N_u)$, while preserving the per-component structure that distinguishes block-wise μP from naïve μP (different scalings per SSM component).

The empirical comparison between the approximated and full block-wise μP, both under AdamW on hybrids, is reported in Appendix A.4: the approximated variant stays within 0.5% of naïve μP at w=1024 after 20k steps on WIKITEXT-103, while the full prescription produces 5.6–7.7% higher loss on every (architecture, width) cell.

A.4 Applying μP-SSM Beyond the Pure-SSM SGD Setting

Chennuru Vankadara et al. [2024] derived their μP-SSM scaling rule (Table 1) for pure SSMs under plain SGD in the limit $N_u, N_x \to \infty$ (with $N_u = 16N_x$ in their reported experiments). Our work uses a different setting along three axes simultaneously: hybrid Transformer-SSM architectures, AdamW, and a fixed $N_x = 16$ as N_u grows with width. This appendix records how the Table 1 rule behaves across the gap, in two settings: (i) on pure Mamba under SGD but with our fixed-N_x setup, and (ii) on our headline hybrid grid under AdamW.

We use a properly anchored implementation of Table 1 (anchored at a reference width N_u^{ref}; we use $N_u^{\text{ref}} = 512$, corresponding to our proxy width w=256):

$$\eta_A(N_u) = \eta_{\text{base}} \cdot \frac{N_u}{N_u^{\text{ref}}} \quad \in \Theta(N_u) \tag{4}$$

$$\eta_B(N_u) = \eta_{\text{base}} \cdot \sqrt{\frac{N_u^{\text{ref}}}{N_u}} \quad \in \Theta(1/\sqrt{N_u}) \tag{5}$$

$$\eta_C(N_u) = \eta_{\text{base}} \cdot \frac{1}{N_x}\sqrt{\frac{N_u^{\text{ref}}}{N_u}} \quad \in \Theta(1/(N_x\sqrt{N_u})) \tag{6}$$

By construction, all three LRs equal η_{base} (modulo the $1/N_x$ factor for η_C) at the reference width N_u^{ref}. At target widths, η_A grows linearly with N_u while η_B and η_C shrink as $1/\sqrt{N_u}$. At w=1024 (N_u=2048) with η_{base}=1.0: η_A=4.0, η_B=0.5, η_C=0.031.

Setting (i): pure Mamba under SGD with fixed N_x. We run the Table 1 rule on pure Mamba under plain SGD (no gradient clipping, 20k steps, WikiText-103) across widths $\{128, 1024\}$ and LRs $\{0.3, 1.0, 3.0, 10.0, 30.0, 100.0\}$. Table 10 reports the results.

Table 10: Faithful block-wise μP vs. naïve μP under plain SGD on pure Mamba (20k steps, WikiText-103). Final training loss.

LR	naïve w=128	block-wise w=128	naïve w=1024	block-wise w=1024
0.3	10.39	8.23	8.83	8.34
1.0	9.52	7.01	7.92	7.43
3.0	7.88	6.64	7.38	6.79
10.0	6.58	6.30	6.88	6.34
30.0	5.96	6.05	6.39	5.95
100.0	5.55	**DIV** (step 942)	6.00	5.96

The Table 1 rule produces lower loss than naïve μP at low-to-mid LRs (0.3–3.0) at both widths, consistent with the rule doing the per-component balancing it was designed for. At the highest LR (η=100), the rule diverges at w=128 (loss $\to$ 5706 at step 942) while naïve μP completes all 20k steps; at w=1024 both converge to similar loss. This differs from the Figure 3 pattern of

Chennuru Vankadara et al. [2024], which reports a wider stability window for μP-SSM than for the μP heuristic.

The proximate cause is the N_u/N_x ratio. Chennuru Vankadara et al. [2024]'s Figure 3 setting holds $N_u = 16N_x$ with both axes scaling together; in our MambaBlock, $N_x = d_{\text{state}} = 16$ is fixed across widths while $N_u = d_{\text{inner}} = 2d_{\text{model}}$ grows with width. So $N_u/N_x = 16$ only at w=128; at w=1024 we operate at $N_u/N_x = 128$. The asymptotic scalings $\Theta(N_u)$ and $\Theta(1/\sqrt{N_u})$ behave differently when N_x is held fixed versus scaled with N_u, and the growing-N_x limit the theorems use does not match our fixed-N_x setup. Testing the rule with N_x growing proportionally (i.e., $N_u = 16N_x$ at every width) closes this regime gap empirically: in a follow-up sweep at $N_u = 16N_x$ across $w \in \{128, 256, 512, 1024\}$ and the same SGD setup, the Table 1 rule produces lower loss than naïve μP at low-to-mid LRs ($\eta \in [0.3, 30]$) at every width, with naïve catching up only at η=100, partially recovering Vankadara's reported loss-ordering claim in the setting of their experiments.

Setting (ii): hybrid Transformer-SSM under AdamW. This is our actual experimental regime. The Table 1 rule was derived without modeling AdamW's per-parameter normalization or the block heterogeneity from non-SSM components, and we observe that its behavior here differs from what Setting (i) suggests. Because the rule is only one possible point of comparison and the headline of the paper is a finding about naïve μP, we summarize the results here rather than carry full per-cell tables in the main text.

*Transfer at 2k steps (*WIKITEXT-2*, Table 1 grid).* The Table 1 rule's optimum LR drifts with width on every SSM-bearing architecture we measured (7 of 7: `MM`, `MTMT`, `MHMH`, `HMHM`, `HHHH`, `HTHT`, plus `JAMBA`): the w=1024, η=3.0 run diverges on all seven, collapsing the optimum to η^*=1.0, while at smaller widths η^*=3.0 is preferred on 5 of 7 archs (`MM`, `MHMH`, `HMHM`, `HHHH`, `JAMBA`; on `MTMT` and `HTHT` a single proxy width prefers η^*=1.0). Only `TT` (pure Transformer, where the SSM-specific rules do not fire) preserves η^*=1.0 across widths. The drift has the same mechanism as in Setting (i): $\eta_A \propto N_u$ becomes large at scale relative to the gradient magnitudes our hybrid blocks produce.

*Convergence at 20k steps (*WIKITEXT-103*).* At the safely-transferable LR η=1.0, block-wise μP produces 5.6–7.7% higher loss than naïve μP across HHHH, MHMH, MTMT $\times$ $w \in \{256, 512, 1024\}$ ($n = 9$ cells, no exceptions; Table 11). The approximated variant, used in the main experiments, stays within 0.5% of naïve at w=1024 (`MHMH`, 20k steps, one seed; 0.44% in training and 0.34% in validation loss); under AdamW the explicit per-component LR rule is therefore redundant with $g/\sqrt{v}$, and the redundancy slows training (Section 5.3).

Table 11: Faithful block-wise μP vs. naïve μP on hybrids under AdamW (20k steps, WIKITEXT-103, η_{base}=1.0, warmup 200 steps). Final training loss; relative gap is $(\mathcal{L}_{\text{block-wise}} - \mathcal{L}_{\text{naïve}})/\mathcal{L}_{\text{naïve}}$.

Arch	Width	Naïve	Block-wise	Gap
`HHHH`	256	3.514	3.712	+5.6%
`HHHH`	512	3.270	3.500	+7.0%
`HHHH`	1024	3.191	3.414	+7.0%
`MHMH`	256	3.741	3.957	+5.8%
`MHMH`	512	3.317	3.520	+6.1%
`MHMH`	1024	3.171	3.395	+7.1%
`MTMT`	256	3.607	3.835	+6.3%
`MTMT`	512	3.495	3.762	+7.7%
`MTMT`	1024	3.158	3.403	+7.7%

Where this leaves μP-SSM and our work. Same-group follow-ups extend the μP framework along axes other than the SSM rule itself (Haas et al. [2024] (μP^2) address sharpness-aware minimization; sparse-architecture extensions appear in Dey et al. [2024], Małaśnicki et al. [2026]), and a side-by-side study under matched N_u/N_x has not yet appeared in the published literature. Production hybrids that deploy μP at scale (Falcon-H1 with 35 per-component multipliers; community Mamba-μP implementations with zero-initialized convolutions) adopt their own per-component recipes targeting production-training stability, a problem distinct from the LR-transfer question we focus on (Section 2.4). Under AdamW on hybrids with fixed N_x, the block-wise μP rule fires at the gradient level (Appendix E.2) but is redundant with Adam's $g/\sqrt{v}$ at training time (Section 5.3).

Our approximated block-wise variant addresses this redundancy by scaling η_A and η_B with the naïve-μP exponent (dropping the $\eta_A \in \Theta(N_u)$ term that AdamW already supplies), preserving the per-component initialization structure while avoiding the double normalization.

Regime gaps from the original derivation. We work outside the regime in which Chennuru Vankadara et al. [2024]'s rules were derived along four axes: (i) hybrid Transformer-SSM blocks vs the pure SSMs of their experiments; (ii) AdamW vs the SGD step from which their Table 1 is derived; (iii) fixed N_x=16 vs growing N_x (N_u/N_x=16 in their reported experiments; proportional scaling of N_x with N_u is impractical at scale because SSM state memory scales as $N_u \times N_x$, and is therefore not used in any production hybrid we are aware of); and (iv) the discretization $B_d \approx \Delta t \cdot B$ used in the official Mamba library [Gu and Dao, 2024] vs the true-ZOH form $B_d = (\Delta t \cdot A)^{-1}(\exp(\Delta t \cdot A) - I) \cdot \Delta t \cdot B$ underlying their Table 1 derivation. The simplified-ZOH choice (iv) is the field standard, shared by Mamba-2 [Dao and Gu, 2024], Jamba [Jamba Team, 2025], Nemotron-H [NVIDIA, 2025], and Falcon-H1 [Zuo et al., 2025], so it is not a quirk of our setup. We apply Vankadara's initialization formulas and (for the non-approximated variant) their LR exponents anchored at our proxy width N_u^{ref}=512; whether the same exponents are theoretically optimal under simplified ZOH is not established. Implementing true-ZOH Mamba and re-evaluating block-wise μP on its own derived discretization, ideally also under proportional N_u/N_x scaling and at production-scale N_x, is left to future work.

Relation to the main text. The "block-wise μP (approximated)" results in Section 4 and Table 1 use our approximated variant (Section 3, Appendix A.3) under AdamW on hybrids with a single shared η_{base}. The full block-wise μP prescription is reported alongside in Table 1 (transfer column) and Table 11 above (convergence comparison); a gradient-level comparison is in Appendix E.2. Our main claim (naïve μP + AdamW achieves 0.0% transfer gap across all 7 architectures) rests on naïve μP's own empirical behavior, not on a comparison to block-wise. In the side-by-side comparisons we do report, naïve achieves lower loss than both block-wise variants under AdamW on hybrids, and the full prescription's optimum additionally drifts with width on every SSM-bearing architecture in the 2k-step transfer grid.

B Unit Interface and Depth Details

This section provides additional details on the unit-interface and depth validation experiments (Section 4.1).

B.1 Max Stable Learning Rate

Table 12 reports the maximum LR at which training stays bounded for each (interface, scheme, width) cell in the unit-interface sweep.

Table 12: Max stable LR at the unit interface across all 9 (T, M, H) ordered pairs (darker = less stable), 1k steps (warmup 50) on WIKITEXT-2. LR grid: $\{0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1.0\}$; an entry of 1.0 indicates stability at the grid maximum (the true max stable LR may be higher). Naïve μP: stable at η=1.0 across every (interface, width) cell. SP: max stable LR shrinks with width. Block-wise μP (approximated): stable across every cell, but with η_A instead set to $\eta_{\text{base}} N_u$ it collapses for serial-SSM interfaces (`TM`, `MT`, `MM`) where the per-component LR correction stacks with Adam's $g/\sqrt{v}$. Block-wise μP (faithful, the literal Vankadara prescription): stable across every cell, in contrast to the approximated variant with $\eta_A = \eta_{\text{base}} N_u$.

		Width (d_{model})				
Interface	Scheme	128	256	512	1024	2048
TT	SP	1.0	0.30	0.30	0.30	0.03
	Naïve μP	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Block-wise (approx.)	1.0	1.0	1.0	1.0	1.0
	Block-wise (faithful)	1.0	1.0	1.0	1.0	1.0
TM	SP	0.30	0.30	0.30	0.10	0.03
	Naïve μP	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Block-wise (approx.)	1.0	1.0	1.0	1.0	1.0
	Block-wise (approx.), $\eta_A = \eta_{\text{base}} N_u$	0.03	0.03	0.01	$\leq$0.003	$\leq$0.003
	Block-wise (faithful)	1.0	1.0	1.0	1.0	1.0
TH	SP	0.30	0.30	0.30	0.10	0.03
	Naïve μP	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Block-wise (approx.)	1.0	1.0	1.0	1.0	1.0
	Block-wise (faithful)	1.0	1.0	1.0	1.0	1.0
MT	SP	1.0	0.30	0.30	0.30	0.10
	Naïve μP	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Block-wise (approx.)	1.0	1.0	1.0	1.0	1.0
	Block-wise (approx.), $\eta_A = \eta_{\text{base}} N_u$	0.03	0.01	0.01	$\leq$0.003	0.01
	Block-wise (faithful)	1.0	1.0	1.0	1.0	1.0
MM	SP	1.0	0.30	0.30	0.30	0.10
	Naïve μP	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Block-wise (approx.)	1.0	1.0	1.0	1.0	1.0
	Block-wise (approx.), $\eta_A = \eta_{\text{base}} N_u$	0.03	0.01	0.01	$\leq$0.003	$\leq$0.003
	Block-wise (faithful)	1.0	1.0	1.0	1.0	1.0
MH	SP	0.30	0.30	0.30	0.30	0.10
	Naïve μP	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Block-wise (approx.)	1.0	1.0	1.0	1.0	1.0
	Block-wise (faithful)	1.0	1.0	1.0	1.0	1.0
HT	SP	1.0	1.0	0.30	0.30	0.10
	Naïve μP	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Block-wise (approx.)	1.0	1.0	1.0	1.0	1.0
	Block-wise (faithful)	1.0	1.0	1.0	1.0	1.0
HM	SP	1.0	0.30	0.30	0.30	0.03
	Naïve μP	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Block-wise (approx.)	1.0	1.0	1.0	1.0	1.0
	Block-wise (faithful)	1.0	1.0	1.0	1.0	1.0
HH	SP	1.0	0.30	0.30	0.30	0.10
	Naïve μP	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Block-wise (approx.)	1.0	1.0	1.0	1.0	1.0
	Block-wise (faithful)	1.0	1.0	1.0	1.0	1.0

B.2 Depth Scaling

Figure 2 shows loss as a function of depth. For all four schemes, loss decreases with depth on both `MHMH` and `HHHH` (panels a and b).

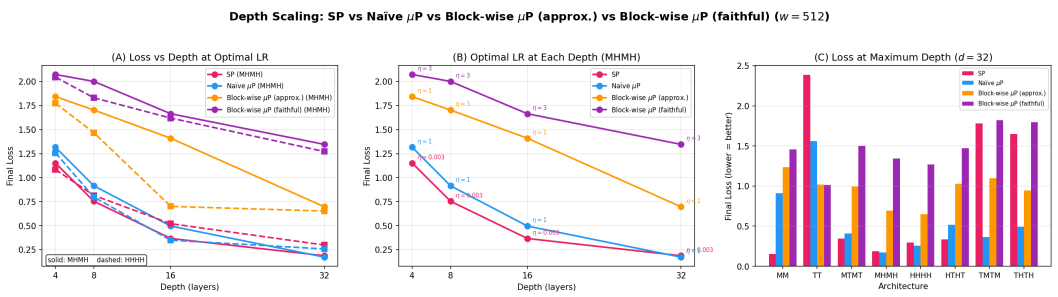


Figure 2: Depth scaling results comparing all four schemes (SP, naïve μP, block-wise μP approximated, block-wise μP faithful) at w=512, 2k steps on WIKITEXT-2. (a) Loss vs. depth at each scheme's optimal LR for `MHMH` and `HHHH`. (b) Loss vs. depth for `MHMH` with the optimal LR labeled at each point. (c) Loss at maximum depth (d=32) for all 8 architectures. All schemes show decreasing loss with depth; the two block-wise variants have higher loss at the 2k-step horizon on WIKITEXT-2; the approximated variant's 20k-step loss on WIKITEXT-103 stays within 0.5% of naïve μP at w=1024 (Appendix A.4). The faithful variant retains a 5–8% loss gap relative to naïve μP even at convergence (Table 11).

C Width Transfer Details

Each full LR-sweep table in this section shows final training loss unless its caption states validation loss (lower is better). Bold values indicate the best LR at each (architecture, width) combination. *div* indicates divergence (loss $>$ 20). All experiments use depth$=16$ unless otherwise noted.

C.1 Joint Depth-Width Transfer

η=1.0 is optimal at the full target (w=1024, d=32) for both `HHHH` and `MHMH`, with 0.0% gap from the proxy model (w=256, d=8, $\sim$11–14$\times$ smaller in parameters). At the proxy horizon, loss at η=1.0 is not monotonic in depth or width across the grid (Figure 3). The full LR sweep across the depth$\times$width grid is in Table 13.

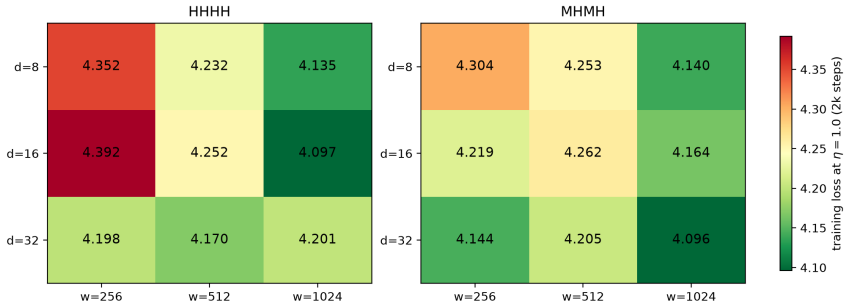


Figure 3: Loss at η=1.0 across the width$\times$depth grid (training loss, 2k steps; values from Table 13). Best: `MHMH` (w=1024, d=32) at 4.096.

Table 13: Depth-width joint transfer for MHMH and HHHH (WIKITEXT-103, 2k steps, naïve μP). Gap = relative loss penalty of using the proxy-transferred LR (η=1) instead of each row's best-of-grid value, $(\mathcal{L}(\eta{=}1) - \mathcal{L}^*)/\mathcal{L}^*$. The 3-point grid does not sample the basin between η=1 and η=3 (Tab 19 shows the true basin lies near $\eta \approx 1.5$–2); cells where η=3 wins by less than 1% should be read as basin-edge sampling artifacts, not as preferring high LR.

Arch	Depth × Width	η=0.3	η=1.0	η=3.0	Gap @ η=1
MHMH	d=8, w=256	4.508	**4.304**	4.318	0.00%
	d=8, w=512	4.488	4.253	**4.230**	0.55%
	d=8, w=1024	4.403	**4.140**	4.204	0.00%
	d=16, w=256	4.411	**4.219**	4.241	0.00%
	d=16, w=512	4.495	**4.262**	4.843	0.00%
	d=16, w=1024	4.453	**4.164**	4.246	0.00%
	d=32, w=256	4.329	**4.144**	4.819	0.00%
	d=32, w=512	4.437	**4.205**	4.305	0.00%
	d=32, w=1024	4.388	**4.096**	4.206	0.00%
HHHH	d=8, w=256	4.527	**4.352**	4.363	0.00%
	d=8, w=512	4.458	**4.232**	4.258	0.00%
	d=8, w=1024	4.413	**4.135**	4.156	0.00%
	d=16, w=256	4.562	**4.392**	4.465	0.00%
	d=16, w=512	4.504	**4.252**	4.359	0.00%
	d=16, w=1024	4.384	**4.097**	4.161	0.00%
	d=32, w=256	4.366	**4.198**	4.293	0.00%
	d=32, w=512	4.364	**4.170**	4.226	0.00%
	d=32, w=1024	4.476	**4.201**	4.288	0.00%

C.2 FineWeb-Edu Width Transfer

We additionally validate width transfer on FINEWEB-EDU (164M training tokens from a 500M-token subset, 20k steps) for HHHH, MHMH, and MTMT under both SP and naïve μP (Table 14); the corresponding validation-loss trajectory at the 5B-token run of Section 4.2 is shown in Figure 4.

Table 14: FineWeb-Edu width transfer (20k steps, 164M training tokens from a 500M-token subset). SP's optimal LR shifts 3× ($0.003 \to 0.001$); transferring the proxy optimal LR incurs 1.1–8.7% penalty (largest on MHMH and MTMT, smallest on HHHH).

		w=256	w=1024		
Arch	Scheme	Loss (opt. LR)	Opt. LR	Transferred	Retuned
HHHH	Naïve μP	4.467 (η=1.0)	1.0	**4.115**	—
HHHH	SP	4.452 (η=0.003)	0.001	4.194	4.149
MHMH	Naïve μP	4.313 (η=1.0)	1.0	**3.993**	—
MHMH	SP	4.315 (η=0.003)	0.001	4.339	3.998
MTMT	Naïve μP	4.451 (η=1.0)	1.0	**4.033**	—
MTMT	SP	4.460 (η=0.003)	0.001	4.403	4.052

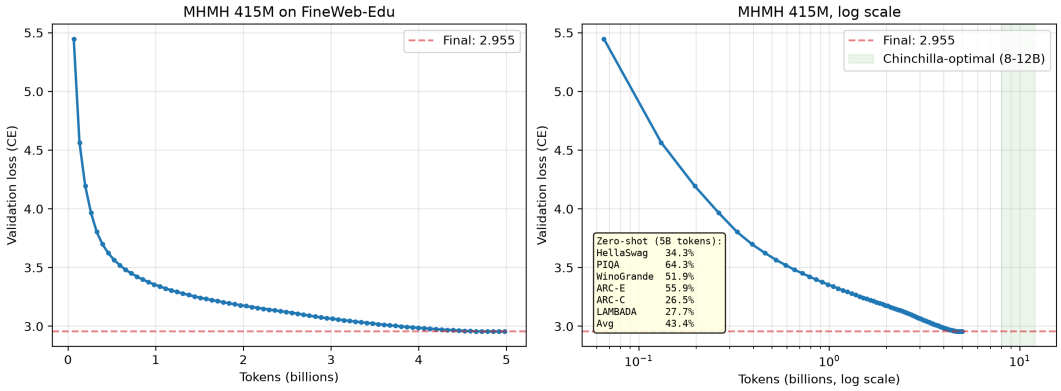


Figure 4: Validation loss curve for MHMH 415M on FINEWEB-EDU (5.0B tokens, 38k steps). Smooth convergence from 5.45 (step 500) to 2.96; η=1.0 transferred from proxy (w=256, 65M params) without retuning.

C.3 Width 2048 and Billion Scale

Extending the transfer range from $4\times$ (256→1024) to $8\times$ (256→2048), we train MHMH (1.24B params) and HHHH (1.66B params) at w=2048. Naïve μP's optimal LR at the 2k-step horizon is η=1.5 for both architectures, consistent with smaller widths (Table 19); at convergence (20k, WIKITEXT-103) the optimum shifts back to η=1.0 (MHMH, Table 17). SP at w=2048 only survives at $\eta \leq 0.003$ (η=0.01 near-diverges to loss > 10); naïve μP beats SP by 0.7% (MHMH) and 2.1% (HHHH) at their respective optimal LRs. Width 2048 for MHMH and HHHH (WIKITEXT-103, 2k steps). Naïve μP sweeps $\eta \in [0.3, 3.0]$ (Table 15); SP sweeps $\eta \in \{0.001, 0.003, 0.01\}$ (Table 16). SP's basin is narrower and shifts to $\eta \approx 0.001$ at this scale, with η=0.01 near-diverging.

Table 15: Width 2048, naïve μP sweep.

Arch	η=0.3	η=0.5	η=0.7	η=1.0	η=1.5	η=2.0	η=3.0	Gap @ η=1
MHMH	4.234	4.070	3.989	3.934	**3.907**	3.984	4.044	0.70%
HHHH	4.480	4.296	4.209	4.161	**4.121**	4.173	4.270	0.97%

Table 16: Width 2048, SP sweep.

Arch	η=0.001	η=0.003	η=0.01
MHMH	**3.935**	4.102	*11.3*
HHHH	**4.211**	4.341	*10.8*

Billion scale, horizon, and seeds. The 2k sweep above uses one seed. The runs below extend MHMH along three axes, each separately: three seeds at w=2048 (1.24B parameters), the full training horizon at the same size, and depth up to 1.88B parameters. Billion scale, the full training horizon, and multiple seeds are thus validated individually rather than jointly, and no single run combines them with a full LR sweep. All three report validation loss.

Three seeds at 20k and 60k steps. Table 17 repeats the sweep with seeds 42, 123, and 456 at 20k steps (164M tokens) and at 60k steps (492M tokens) on WIKITEXT-103. At both horizons η=1.0 is the argmin at every seed, and no run diverges (0/24). At 20k this is also the w=256 proxy's optimum (Table 26), so the transfer gap is 0.0%; across seeds the loss at η=1.0 varies by a coefficient of variation of about 0.1%, and η=1.5 is 0.289 nats worse on the seed mean. At 60k, we observe that the training loss favors $\eta \leq 0.7$ while the validation loss still selects η=1.0; we conjecture that this reflects these runs passing over WIKITEXT-103 about four times, and read the optimum from validation loss.

Table 17: MHMH at w=2048 (1.24B parameters) on WIKITEXT-103, three seeds: validation loss at 20k and 60k steps. Bold marks the argmin of each column; – marks an LR not run at that horizon.

	20k steps (164M tokens)				60k steps (492M tokens)			
η	seed 42	seed 123	seed 456	mean	seed 42	seed 123	seed 456	mean
0.5	–	–	–	–	3.290	3.284	3.275	3.283
0.7	3.234	3.237	3.238	3.236	3.260	3.257	3.256	3.258
1.0	**3.223**	**3.231**	**3.229**	**3.228**	**3.215**	**3.227**	**3.220**	**3.220**
1.5	3.636	3.415	3.500	3.517	3.315	3.301	3.254	3.290
2.0	3.661	3.563	3.481	3.568	–	–	–	–

Chinchilla-horizon run. Without retuning, the same model is trained at η=1.0 on FINEWEB-EDU for 190k steps at sequence length 2048 and effective batch 64, which is 24.9B tokens or about 20 tokens per parameter. It converges without divergence to a validation loss of 2.577, with a zero-shot average of 0.50 over HellaSwag, PIQA, WinoGrande, ARC-Easy, ARC-Challenge, and LAMBADA (normalized accuracy where reported, accuracy otherwise). This is one LR and one seed, so it shows that the transferred LR trains to the full horizon; it does not measure a transfer gap there.

Depth to 1.88B parameters. At w=1024, MHMH at depth 48, 64, 96, and 128 (834M to 1.88B parameters) is swept over the fine-grained 7-point grid for 2k steps with effective batch 64 and one seed (Table 18). η=1.0 is the argmin at depth 48 and 64 and stays within 0.043 nats of the best grid point at depth 96 and 128. With one seed and a short horizon, this bounds the cost of reusing η=1.0 rather than establishing a 0% gap.

Table 18: MHMH at w=1024 and depth 48–128 (WIKITEXT-103, 2k steps, effective batch 64, one seed, validation loss). η^* is the best of $\{0.3, 0.5, 0.7, 1.0, 1.5, 2.0, 3.0\}$; Δ is the loss of η=1.0 above it.

Depth	Parameters	η^*	Δ (nats)	Δ (%)
48	834M	1.0	0.000	0.00%
64	1.04B	1.0	0.000	0.00%
96	1.46B	0.7	0.043	1.06%
128	1.88B	1.5	0.004	0.09%

C.4 Fine-Grained LR Sweep

To rule out discrete-grid artifacts in our 0.0% gap claim, we fill in intermediate LR points ($\eta \in \{0.5, 0.7, 1.5, 2.0\}$) to produce 7-point loss-vs-LR curves at each width. The curves superimpose across widths for both MHMH and HHHH. At this 2k-step horizon the optimum is η=1.5 at every width, and η=1.0 costs at most 0.6% (Table 19). The transfer gap from proxy (w=256) to target (w=1024) is 0.0% for both architectures at the grid optimum; not a grid artifact. At 20k steps (HHHH; the intermediate points use a 1000-step warmup), the optimum is η=1.0 at all three widths with a basin spanning $\eta \in [0.5, 1.5]$ (per-cell spread $\leq$0.7%), and at η=2.0 the w=1024 run becomes unstable (229 spikes, final loss 3.80), localizing the upper edge of the basin to $\eta \approx 1.5$ at the largest width. The full 7-point sweep at all 3 widths for both architectures is in Table 19.

Table 19: 7-point loss vs. LR for MHMH and HHHH (WIKITEXT-103, 2k steps, naïve μP). Gap = $(\mathcal{L}(\eta{=}1) - \mathcal{L}^*)/\mathcal{L}^*$ in %, the relative penalty of using the proxy-transferred LR (η=1) instead of the row's best-of-grid value.

Arch	Width	η=0.3	η=0.5	η=0.7	η=1.0	η=1.5	η=2.0	η=3.0	Gap @ η=1
MHMH	256	4.411	4.290	4.262	4.218	**4.202**	4.209	5.523	0.38%
	512	4.495	4.354	4.301	4.265	**4.240**	4.249	4.255	0.59%
	1024	4.453	4.285	4.221	4.170	**4.145**	**4.145**	4.264	0.60%
HHHH	256	4.562	4.468	4.421	4.403	**4.387**	4.399	4.443	0.36%
	512	4.504	4.368	4.294	4.268	**4.255**	4.286	4.349	0.31%
	1024	4.386	4.217	4.142	4.107	**4.083**	4.126	4.174	0.59%

C.5 Architecture Variants

C.5.1 Mamba-2 Compatibility

Mamba-2 blocks substituted into the `MHMH` layout (WIKITEXT-103, 2k steps). Naïve μP sweeps the high-LR basin ($\eta \in [0.5, 5.0]$, Table 20); SP sweeps the low-LR basin ($\eta \in [0.0003, 0.03]$, Table 21). The naïve μP basin minimum is at η=2.0 at both widths, with η=3.0 tied at w=256; η=5.0 is past the basin. SP optimal LR shifts 10× from w=256 (η=0.01) to w=1024 (η=0.001), a larger shift than Mamba-1's 3× (Table 22); η=0.03 near-diverges at w=1024.

Table 20: Mamba-2 `MHMH`, naïve μP sweep.

W	η=0.5	η=0.7	η=1.0	η=1.5	η=2.0	η=3.0	η=5.0	Gap @ η=1
256	4.190	4.142	4.113	4.117	**4.093**	4.096	4.318	0.48%
1024	4.360	4.299	4.253	4.235	**4.232**	4.332	4.480	0.51%

Table 21: Mamba-2 `MHMH`, SP sweep.

W	η=0.0003	η=0.001	η=0.003	η=0.01	η=0.03
256	5.163	4.541	4.247	**4.198**	4.825
1024	4.624	**4.342**	4.384	4.580	*11.3*

C.5.2 SP Fair Baseline

Table 22: SP fair baseline (WIKITEXT-103, 20k steps).

	η=0.0003	η=0.001	η=0.003	η=0.01	η=0.03
MHMH w=256	4.181	3.826	**3.774**	3.858	4.505
MHMH w=1024	3.267	**3.210**	3.635	3.853	*11.5*
MTMT w=256	4.004	3.685	**3.615**	3.671	4.462
MTMT w=1024	3.226	**3.186**	3.559	*11.5*	*11.5*

C.5.3 Gated Linear Attention

To test a linear-recurrent mixer outside the Mamba family, we replace the Mamba block with a Gated Linear Attention (GLA) layer [Yang et al., 2024] and interleave it with Transformer blocks (`GTGT`, depth 16). Every GLA weight is a real-valued, fan-in-scaled projection, so naïve μP applies to it as to the Transformer blocks. Under naïve μP on WIKITEXT-103 (20k steps, one seed), η=1.0 is the argmin at every width on both training and validation loss, so the transfer gap across this fourfold width range is 0.0% (Table 23); no run diverges.

Table 23: `GTGT` (GLA interleaved with Transformer blocks, depth 16) under naïve μP on WIKITEXT-103 (20k steps, one seed): validation loss. Bold marks the argmin of each row.

Width	Parameters	η=0.3	η=1.0	η=3.0	η^*
256	60M	3.732	**3.708**	3.785	1.0
512	137M	3.575	**3.556**	3.587	1.0
1024	342M	3.488	**3.450**	3.501	1.0

C.6 Nemotron-H Block Composition

Section 4.3 reports an LR-transfer experiment on a Nemotron-H [NVIDIA, 2025] reproduction, but the headline architecture set used in the rest of the paper (T, M, H blocks) does not include the block types Nemotron-H is built from. For reproducibility, we record the block-type letters our implementation uses and the layer-pattern they produce.

Block types. Nemotron-H uses three block types, which we expose via single-letter codes consistent with the production paper:

- **N**: Mamba-2 layer [Dao and Gu, 2024] (SSD-style selective scan; requires the `mamba_ssm` CUDA kernel).
- **F**: FFN layer (squared-ReLU MLP only, no attention or SSM in this layer).
- **A**: attention-only layer that replaces an N in an N-F pair when sparse attention is inserted.

This is distinct from our headline T (Transformer block: GQA + SwiGLU FFN, fused) and M (Mamba-1) blocks: Nemotron-H separates the FFN into its own F layer instead of fusing it with attention or SSM, and inserts a small fraction of attention-only A layers rather than full Transformer blocks.

Layer pattern. The base pattern alternates N and F at every depth (Mamba-2 followed by MLP). A small fraction of attention layers (default α=0.08) is dispersed evenly by replacing an N with an A in some N-F pairs; the first layer is always N and the last is always F. At the depths we test:

depth 16 : `NFNFNFNFAFNFNFNF` (7 N + 8 F + 1 A; 6.2% attention)

depth 32 : `NFNFNFNFNFNFNFNFAFNFNFNFNFNFNFNF` (15 N + 16 F + 1 A; 3.1% attention)

The attention fraction we observe (3–6%) is slightly below the $\sim$8% target because rounding to integer attention positions takes precedence over the exact ratio at small depth.

Deviations from the production specification. Our Nemotron-H reproduction matches the published model on the load-bearing components: Mamba-2 SSD blocks, squared-ReLU FFN, no positional encoding in attention, and an $\sim$8% attention-layer density (rounded to integer positions per depth as above). We tested two attention implementations to bracket the production model:

- **Multi-head attention (MHA) variant.** Our initial reproduction used full multi-head attention ($n_{\text{kv}} = n_{\text{heads}}$) without rotary position embeddings. This is simpler than the production architecture but probes the same N/F/A composition.
- **Precise GQA variant.** A second reproduction uses grouped-query attention at ratio 4 ($n_{\text{kv}} = n_{\text{heads}}/4$) with no rotary embeddings, matching the published Nemotron-H attention configuration. This is the implementation reported in Section 4.3 and Table 3.
- **Dense attention pattern (no sliding window or other sparsification).** The A-layers are full dense self-attention in both variants. Nemotron-H reports sparse attention at production scale; the relevant μP analysis for sparse architectures is in Dey et al. [2024], who show that sparsity-aware μP rules generalize the standard prescription. Our finding (0.0% LR transfer gap on Nemotron-H under naïve μP) is therefore a result on the dense-attention reproduction and does not yet probe the sparse-pattern regime; we expect transfer to extend, but flag this as untested.
- **Smaller SSM state dimension.** Our reproduction uses N_x=16 across all widths, while the production Nemotron-H 8B model uses N_x=128 in its Mamba-2 SSD blocks. Our small-scale validation tests the LR-transfer mechanism rather than reproducing the production model exactly; the field-standard N_x=16 choice (also used in Mamba-1, Jamba, and Falcon-H1) keeps our setup comparable across architectures; the released-configuration runs below use N_x=128.

Released-configuration dimensions. Against the released Nemotron-H-8B configuration, our reproduction matches 10 of 14 structural properties (MLP activation, Mamba-2 head dimension and expansion, convolution kernel, GQA ratio, positional encoding, embedding tying, first- and last-layer types, and attention-layer fraction, within rounding as above) and differs in four fixed dimensions: the MLP intermediate ratio (2.75 vs 5.25), the SSM state (16 vs 128), the number of Mamba-2 groups (1 vs 8), and the attention head dimension (32 vs 128). None of them scales with width. With these four set to the released values, and width and depth kept at our scale, naïve μP selects η^*=1.0 at widths 512, 1024 and 2048 on both training and validation loss, and no run diverges (Table 24).

Table 24: Nemotron-H with the released Nemotron-H-8B configuration's MLP ratio, SSM state, Mamba-2 groups and attention head dimension (naïve μP, depth 16, WIKITEXT-103, 20k steps, warmup=1000). Final training loss; validation loss selects the same η^* at every width, and the transfer gap from w=512 is 0.0%.

Width (params)	η=0.3	1.0	3.0	5.0	10.0
512 (144M)	3.529	**3.467**	3.498	3.533	3.614
1024 (355M)	3.271	**3.209**	3.223	3.259	3.302
2048 (980M)	3.113	**3.027**	3.052	3.061	3.086

Longer training. At the transferred η=1.0, we trained the depth-16 reproduction of Table 3 on FINEWEB-EDU at about 20 tokens per parameter, with batch 64 and sequence length 128: 57M parameters on 1.15B tokens (w=256, 140k steps) and 299M parameters on 6.0B tokens (w=1024, 733k steps). Neither run diverges; the final validation losses are 4.330 and 3.742.

MHA vs GQA: transfer property holds in both. Both attention implementations show $\sim$0% LR transfer gap under naïve μP, but the optimum LR and absolute loss differ. Table 25 compares the two on the same (depth, width) grid.

Table 25: Nemotron-H attention-implementation comparison (naïve μP, WIKITEXT-103; MHA rows 2k steps with warmup=100, GQA rows 20k steps with warmup=1000). Both variants achieve clean LR transfer; absolute losses and the optimum LR are not comparable across the two horizons.

Variant	Depth	Best LR	w=256	w=512	w=1024	Gap
MHA, no RoPE	16	3.0	4.219	4.120	**4.151**	**0.07%**
MHA, no RoPE	32	3.0	4.453	4.113	**4.067**	**0.00%**
GQA, no RoPE	16	1.0	3.760	3.344	**3.369**	**0.00%**
GQA, no RoPE	32	1.0	3.527	3.313	**3.138**	**0.00%**

The GQA rows are 20k-step runs and the MHA rows 2k-step runs, so their absolute losses and optimum LRs are not directly comparable. Both implementations preserve the headline transfer property: naïve μP with AdamW achieves $\sim$0% transfer gap independent of the attention configuration.

Why this is the right test for the paper's claim. Nemotron-H sits outside our headline grid (different block types, different per-layer composition, different attention placement strategy), so a successful 0.0% LR transfer gap on either reproduction is evidence that naïve μP with AdamW transfers across genuinely independent hybrid designs, not just within the T/M/H block family our main experiments enumerate.

D Robustness Checks

D.1 Convergence at 20k Steps

The 0.0% transfer gap holds at convergence (20k steps on WikiText-103, Table 26). Figure 5 shows a mid-training transient (steps 3k–14k for width 256, shorter for wider architectures) where η=0.3 briefly leads, which is self-correcting. SP divergence rates *increase* with training duration, reaching 100% on all three measured architectures (`HHHH`, `MHMH`, `MTMT`) at 20k steps on the 3-LR$\times$3-width grid.

Table 26: Convergence validation (20k steps, WIKITEXT-103, naïve μP). Optimal LR is η=1.0 at all 9 (arch, width) conditions. SP divergence rate is on the 3-LR $\times$ 3-width grid ($\eta \in \{0.3, 1.0, 3.0\}$, $w \in \{256, 512, 1024\}$, 9 runs per arch); rates rise with training duration.

	Loss at η=1.0			SP divergence rate	
Arch	w=256	w=512	w=1024	@2k	@20k
HHHH	3.514	3.270	3.191	67%	**100%**
MHMH	3.741	3.317	3.171	67%	**100%**
MTMT	3.607	3.495	3.158	78%	**100%**

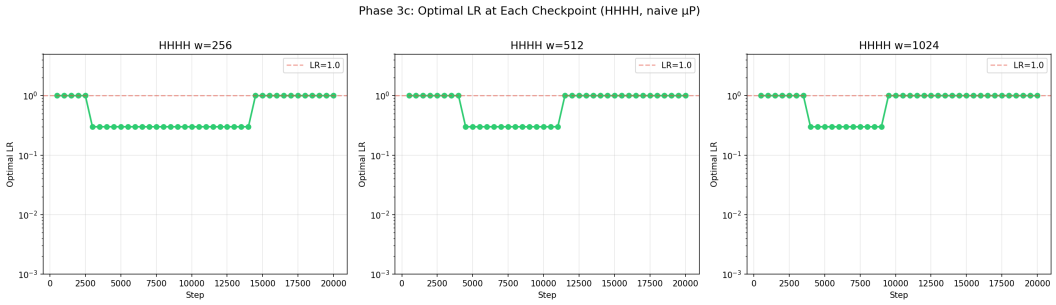


Figure 5: Optimal LR at each training checkpoint (20k steps, WikiText-103). At w=256, η=0.3 briefly leads during mid-training (steps 3k–14k) before η=1.0 recovers. By convergence, η=1.0 is universally optimal.

D.2 Multi-Seed Validation

To show that the 0.0% gap is not a lucky draw, we repeat the width transfer experiment across 3 seeds for both HHHH and MHMH (36 configs: 2 archs $\times$ 2 widths $\times$ 3 LRs $\times$ 3 seeds). LR=1.0 is optimal in 12/12 per-seed conditions (seeds 42, 123, 456). Transfer gap was 0.0% at every seed. Cross-seed CV $\leq$ 1.7% at the optimum (mean 1.2% across the four (arch, width) cells), showing that the results are highly reproducible (Figure 6).

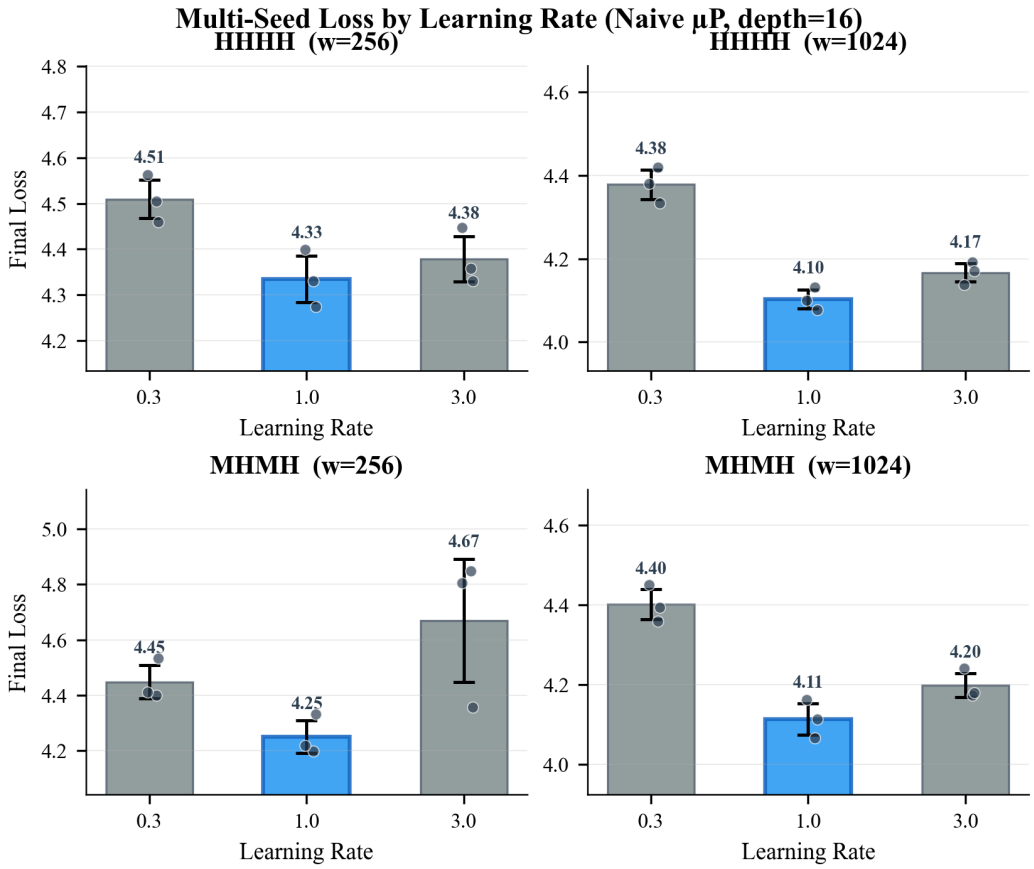


Figure 6: Final training loss (mean ± std) across 3 seeds (2k steps, WIKITEXT-103). η=1.0 (blue) is optimal in all 12 per-seed conditions. Cross-seed CV $\leq$ 1.7% at the optimum, confirming high reproducibility. MHMH η=3.0 at w=256 shows anomalously high variance (CV $\approx$ 5.8%).

D.3 Batch Size Robustness

We wanted to check if the LR transfer depends on batch size or not on the hybrid architectures. We sweep 7 LRs at batch sizes $\{8, 16, 32, 64\}$ for both widths (256, 1024) and both architectures (Table 27). Smaller batches shift the optimal LR downward (from $\eta\approx$1.5 at b=64 to $\eta\approx$0.5–0.7 at b=8), but this shift is *width-independent*; learning rate transfer is still preserved at all batch sizes.

Table 27: Width transfer gap at each batch size (naïve μP, 2k steps, WIKITEXT-103, training loss). Gap: loss at w=256 optimal LR applied to w=1024 vs. best at w=1024.

Batch size	MHMH gap	HHHH gap	MHMH optimal LR	HHHH optimal LR
8	0.5%	0.0%	0.7–1.0	0.5
16	0.0%	1.1%	1.0	0.7–1.0
32	0.3%	0.0%	1.5–2.0	1.0
64	0.0%	0.0%	1.5–2.0	1.5

All gaps are $\leq$1.1%. The practical implication: batch size shifts the optimal LR uniformly across widths, so practitioners can change batch size and re-tune at proxy scale; the transferred LR remains valid at target scale.

D.4 Sequence Length Scaling

72 configs: HHHH/MHMH/TT $\times$ $w \in \{256, 1024\}$ $\times$ 3 LRs $\times$ 4 seq_lens (128–1024) under naïve μP. Batch size adjusted to keep tokens/step constant at 8192. η=1.0 is optimal across all sequence lengths 128–1024, all 3 architectures, and both widths (24/24 conditions); zero divergence across 72 configs. Extending the sweep to seq=2048, the length of the 5.0B- and 24.9B-token runs, adds 18 runs and gives the same result: η=1.0 is optimal in all 6 (architecture, width) cells on both training and validation loss, the width-transfer gap is 0.0%, and none of the 90 runs diverges. Final loss at the optimum is reported in Table 28; MHMH leads at seq=1024, consistent with SSM's long-context advantage.

Table 28: Sequence length scaling at η=1.0 (naïve μP, 2k steps, WIKITEXT-103). Final loss at each (arch, w, seq_len). η=1.0 is optimal at every cell; LR transfer is sequence-length invariant for tested lengths 128–1024.

Arch	Width	seq=128	seq=256	seq=512	seq=1024
HHHH	w=256	4.399	4.100	**4.074**	4.477
	w=1024	4.102	4.262	4.215	4.509
MHMH	w=256	4.213	4.388	4.580	**4.084**
	w=1024	4.161	4.323	4.300	4.338
TT	w=256	4.258	4.235	4.454	4.267
	w=1024	4.255	4.459	4.207	4.547

LR sensitivity across sequence length follows the same architecture ranking we observe at fixed seq_len: HHHH stays within 0.01–0.10 between η=1.0 and η=3.0, MHMH within 0.02–1.4 (one outlier at w=1024, seq=256), and TT spans 0.41–0.70, $\sim 10\times$ more sensitive than HHHH; hybrid architectures degrade more gracefully under LR misspecification.

D.5 T:M Ratio Transfer

We test whether naïve μP LR transfer holds across different Transformer-to-SSM parameter ratios within H blocks. The `ssm_expand` parameter controls $d_{\text{inner}} = d_{\text{model}} \times$ `expand`, which determines the Mamba sub-block size relative to the Transformer sub-block; the resulting per-block parameter breakdown is shown in Table 29.

Table 29: H-block parameter breakdown by `ssm_expand` ($d_{\text{model}} = 512$).

`expand`	d_{inner}	Attention	SwiGLU FFN	T total	M total	T:M Ratio	H block
1	512	1.05M	2.16M	3.21M	0.85M	3.8:1	4.06M
2	1024	1.05M	2.16M	3.21M	1.69M	1.9:1	4.91M
4	2048	1.05M	2.16M	3.21M	3.39M	0.9:1	6.60M

We test 3 uniform ratios plus 3 heterogeneous per-layer patterns (`het_alt`, `het_taper`, `het_reverse`). 252 configs total: 6 patterns $\times$ 3 widths $\times$ 2 schemes $\times$ 7 LRs. Naïve μP achieves 0.0% transfer gap for all 6 configurations (Figure 7).

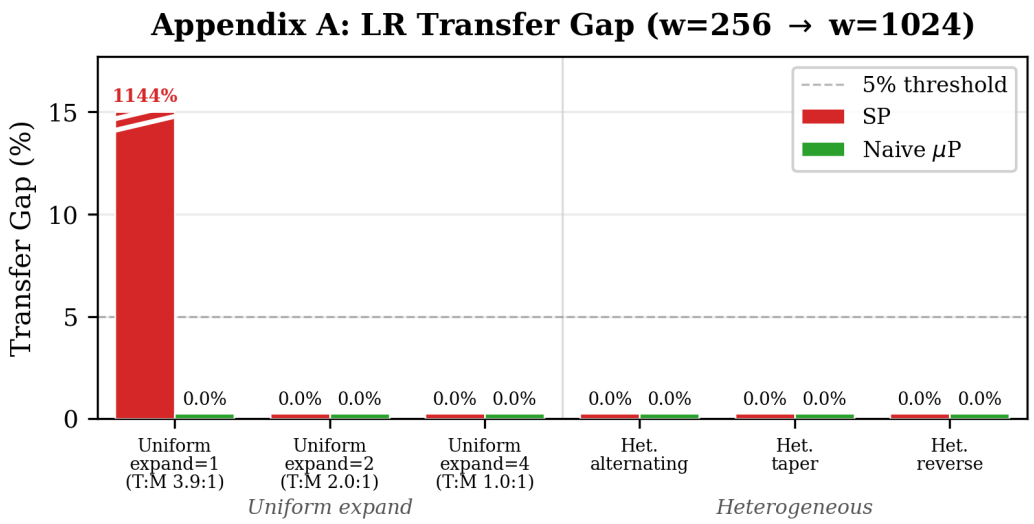


Figure 7: Transfer gap across T:M ratios (naïve μP and SP, WIKITEXT-2, 2k steps, warmup 100; training loss). Naïve μP: 0.0% gap for all 6 configurations. SP: 0.0% for 5/6 but catastrophic (1144%) for `expand1` where the optimal LR shifts with width. LR=1.0 is invariant to the T:M ratio; no re-tuning needed when changing `ssm_expand`.

Note on SP transferring at $5/6$ ratios. SP appears to transfer cleanly on five of the six configurations because over the tested width range its optimum happens to coincide at the proxy and target widths for those particular layouts; this is a feature of the limited grid here rather than a contradiction of our broader finding. The architecture-spanning width-transfer sweep (Table 1) confirms SP's optimum drifts 3–10$\times$ on every arch we tested. The `expand=1` cell is the critical case for this appendix: the SP optimum genuinely shifts with width, producing the 1144% catastrophic gap, while naïve μP stays at η=1.0.

`het_taper` (T-dominant early $\rightarrow$ SSM-heavy late) achieves the best loss (0.114), beating uniform `expand=2` (0.130) by $\sim$12%. This suggests more SSM capacity in late layers is architecturally beneficial.

D.6 Serial T:M Ratio Sweep

Section D.5 covers parallel imbalance inside an H block; this subsection covers *serial* imbalance, where Transformer and Mamba blocks are interleaved at varying ratios along the depth axis. We sweep six fixed-depth (d=16) ratios spanning the full T–M spectrum: `TT` (0% M), `M1T3` (`MTTTMTTTMTTTMTTT`, 25% M), `MTMT` (50% M), `M3T1` (`MMMTMMMTMMMTMMMT`, 75% M), Jamba (`MMMMMMMTMMMMMMMT`, 87.5% M), and `MM` (100% M). Each is run at $\eta \in \{0.003, 0.01, 0.03, 0.1, 0.3, 1.0, 3.0\}$ across $w \in \{256, 512, 1024\}$ under SP, naïve μP, approximated block-wise μP, and the full block-wise prescription (2k steps on WIKITEXT-2).

Table 30: Serial T:M ratio transfer at depth 16. Optimal η^* at each (scheme, pattern, width); shading mirrors Table 1, where uniform color within a row indicates clean LR transfer. Naïve μP holds η^*=1.0 across the full ratio range; approximated block-wise μP drifts on the SSM-bearing layouts; SP drifts on every layout where the SSM density is non-trivial. The full block-wise prescription shifts on every SSM-bearing layout, with η=3.0 diverging at w=1024 on each.

		Naïve μP			Block-wise μP (approx.)			Block-wise μP			SP		
Pattern	%M	w=256	w=512	w=1024	w=256	w=512	w=1024	w=256	w=512	w=1024	w=256	w=512	w=1024
`TT`	0%	**1.0**	**1.0**	**1.0**	1.0	1.0	1.0	1.0	1.0	1.0	0.003	0.003	0.003
`M1T3`	25%	**1.0**	**1.0**	**1.0**	1.0	3.0	3.0	1.0	3.0	1.0	0.003	0.003	0.003
`MTMT`	50%	**1.0**	**1.0**	**1.0**	1.0	1.0	1.0	1.0	3.0	1.0	0.010	0.003	0.003
`M3T1`	75%	**1.0**	**1.0**	**1.0**	1.0	1.0	3.0	1.0	3.0	1.0	0.010	0.003	0.003
`JAMBA` (1:7)	87.5%	**1.0**	**1.0**	**1.0**	1.0	3.0	3.0	3.0	3.0	1.0	0.010	0.003	0.003
`MM`	100%	**1.0**	**1.0**	**1.0**	1.0	3.0	3.0	3.0	3.0	1.0	0.010	0.003	0.003
Transfers?		**Yes** (0/6 shift)			Partially (4/6 shift)			Drifts (5/6 shift)			No (4/6 shift)		

Faithful block-wise μP on this grid: the literal Vankadara prescription shifts on every SSM-bearing layout, as in the width-transfer table (Table 1). At w=1024, η=3.0 diverges on each of them, so the optimum falls to η=1.0; at smaller widths it is η=3.0 (Jamba and MM at $w \leq 512$, M1T3, MTMT, and M3T1 at w=512).

Approximated block-wise μP across the ratio range. At the proxy width w=256, η^*=1.0 for every ratio, matching naïve. The drift only emerges at larger widths: TT and MTMT stay at η^*=1.0 throughout; M3T1 shifts only at w=1024; M1T3, Jamba, and MM shift starting at w=512. Because the proxy is at η=1.0, transferring it to the target width still trains, but the residual loss penalty grows with target SSM density. Table 31 reports the absolute loss at the transferred η=1.0 versus the per-cell optimum for the SSM-bearing patterns at w=1024: penalties stay near 0% for TT and MTMT, climb to 4.2% on M1T3, and reach 24–36% on the heavily SSM-dominated M3T1, Jamba, and MM layouts.

Table 31: Approximated block-wise μP residual loss penalty at w=1024 (WIKITEXT-2, 2k steps). Penalty is the fractional loss increase incurred by transferring the proxy optimum η=1.0 instead of using each row's per-cell optimum at w=1024. Naïve μP has 0% penalty on every row.

Pattern	%M	Loss at η=1.0	Best loss (best η)	Penalty
TT	0%	1.048	1.048 (η=1.0)	0.0%
M1T3	25%	0.755	0.725 (η=3.0)	4.2%
MTMT	50%	0.734	0.734 (η=1.0)	0.0%
M3T1	75%	0.818	0.600 (η=3.0)	36.4%
JAMBA	87.5%	0.852	0.630 (η=3.0)	35.2%
MM	100%	1.144	0.916 (η=3.0)	24.8%

Faithful block-wise μP on Jamba. The faithful Vankadara prescription [Chennuru Vankadara et al., 2024] drifts on Jamba at 2k steps, mirroring the pattern seen on every other SSM-bearing architecture in Section C: η^*=3.0 at $w \leq 512$ shifts to η^*=1.0 at w=1024 as $\eta_A \propto N_u$ destabilizes training (the w=1024, η=3 run diverges). Compared to the approximated variant whose drift onset is partial (T-dominant ratios still transfer cleanly), the full prescription shifts on *every* ratio with non-trivial SSM density, consistent with the observation in Table 1 that block-wise μP's η^* does not transfer cleanly on any SSM-bearing layout.

SP at any imbalance. SP's optimum drifts on 4 of the 6 ratios (MTMT, M3T1, Jamba, MM); on the pure-T and 25%-M layouts (TT, M1T3) the SP optimum happens to coincide between proxy and target widths over our grid, but this is a feature of the local optimum rather than a transfer property (Table 1 shows SP drifts on every architecture once the LR grid extends below η=0.003). For Jamba specifically, the proxy LR (η=0.01 at w=256) produces catastrophic loss at w=1024 where η^*=0.003.

Combined with the parallel-imbalance evidence in Section D.5, this covers every serial T:M ratio used by the production hybrids of Section 2.4.

D.7 Gradient Clipping Threshold

We use grad_clip=1.0 throughout the main experiments and verify that the LR transfer claim does not depend on this specific threshold. We sweep grad_clip $\in \{\text{off}, 0.5, 1.0, 5.0\}$ on MHMH at $w \in \{256, 1024\}$ and $\eta \in \{0.5, 1.0, 1.5\}$ for 2k steps on WIKITEXT-103 (24 configs). At the μP optimum η=1.0, the final loss varies by ≤ 0.020 nats across clip values at both widths, and the optimal LR is unchanged. Disabling clipping entirely does not destabilize training (max-grad-norm trajectories stay below $\sim$5 throughout), confirming that gradient clipping is not load-bearing for the LR-transfer property and that the headline claim is robust to this hyperparameter.

D.8 Where Naïve μP Transfer Does Not Extend

The headline claim of 0.0% LR transfer gap holds for naïve μP + AdamW across the architectures, widths, and depths in our main grid. This subsection documents four boundary conditions where the transfer property weakens or does not apply. Each case fits the two-condition decomposition of Section 5.2; together they sharpen, not contradict, the headline claim.

(i) Standard parameterization. SP is the baseline against which μP was originally designed; without per-component re-scaling, the optimum LR shifts 3–10$\times$ with width across the architectures we tested (Appendix C). On the imbalanced Jamba layout (Section D.6), the SP optimum drifts from η=0.01 at w=256 to η=0.003 at w=1024, with the proxy LR producing catastrophic loss at the target width. This is the regime μP was designed to fix; our headline claim is measured against it.

(ii) Approximated block-wise μP on SSM-heavy serial layouts. On the serial T:M sweep (Section D.6, Table 30), the approximated optimum drifts by one grid step at $w \geq 512$ for SSM-bearing layouts (`M1T3`, Jamba, `MM`; `M3T1` only at w=1024), staying at η^*=1.0 on `TT` and `MTMT`. Reusing the proxy η=1.0 at w=1024 incurs a residual loss penalty of 4–36% on the SSM-dominated layouts (Table 31); the same parameterization transfers cleanly with 0% gap on the balanced `MTMT` layout in the main results.

(iii) Faithful block-wise μP under AdamW at scale. The SSM-specific rules of Chennuru Vankadara et al. [2024] were derived and validated for pure SSMs trained with SGD in the infinite-width limit. On hybrids at 20k steps on WIKITEXT-103, the faithful prescription produces uniformly higher loss than naïve μP across the architectures and widths we measured: +5.6 to +7.7% on HHHH, MHMH, and MTMT at $w \in \{256, 512, 1024\}$ (full table in Appendix A.4). At the 2k-step transfer horizon, its optimum LR also drifts with width on every SSM-bearing architecture (η^*=1 at w=1024 across all 6, with η=3 diverging at that width on each); only pure-Transformer (`TT`) transfers cleanly. The approximated variant used in the main experiments stays within 0.5% of naïve μP at w=1024 by dropping the $\eta_A \in \Theta(N_u)$ term that AdamW's $g/\sqrt{v}$ already supplies (Section 5.3, Appendix A.3).

(iv) Optimizers without per-parameter normalization. Plain SGD does not exhibit the same transfer behavior even when μP is applied (full optimizer ablation in Section E.3). This localizes the LR-transfer property of naïve μP + AdamW to the per-parameter-normalization mechanism ($g/\sqrt{v}$), which matches the two-condition decomposition: μP alone does not deliver transfer in our hybrid setting without an optimizer that supplies Condition 2.

Naïve μP + AdamW transfers cleanly because μP supplies the global condition and AdamW supplies the local one. Adding component-specific scaling on top does not help at the widths and budgets we test.

E Analysis Details

E.1 Coordinate Check

Standard μP validation: train for 10 steps at 5 widths (128, 256, 512, 1024, 2048) with 5 seeds and record mean absolute activation per layer. Under correct μP, lines should overlap; under SP, they diverge.

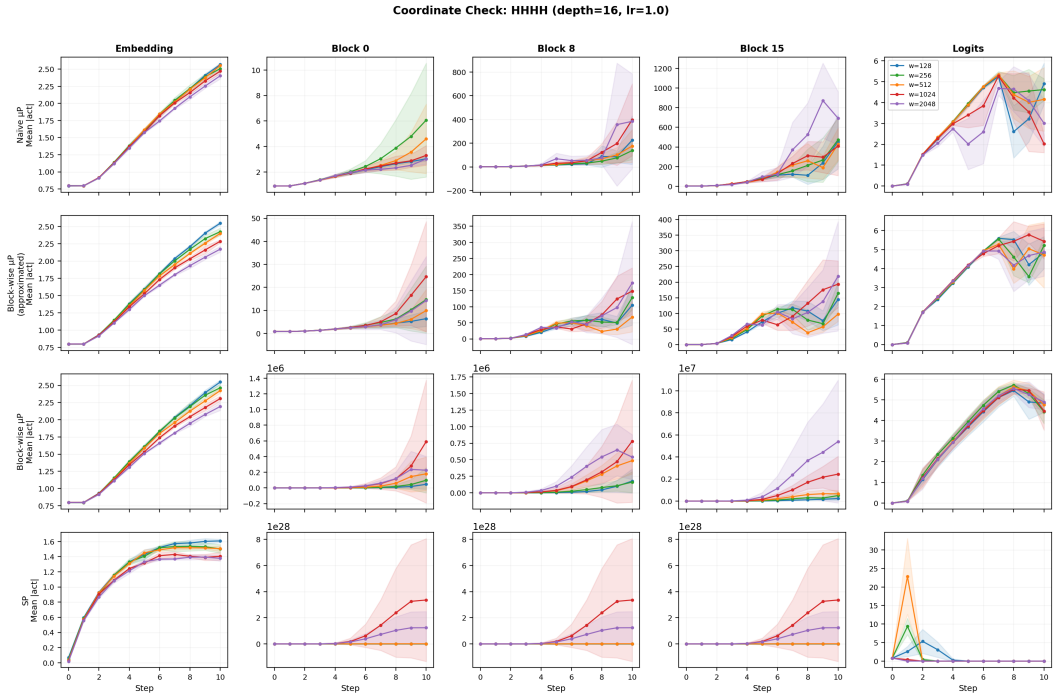


Figure 8: Coordinate check for HHHH (d=16, widths 128–2048). Row 1: naïve μP, hidden activations vary with width (median block width-spread 2.78×, Table 32; violates textbook μP). Row 2: block-wise μP (approximated), similar pattern. Row 3: block-wise μP (literal Vankadara Table 1), hidden activations predominantly 10^3–$10^4\times$ larger in absolute magnitude than naïve at deeper blocks, with the largest width-spread. Row 4: SP, activations explode. Despite the hidden-state divergence, logit-level magnitudes (rightmost column) are comparable across all μP variants (2.0–5.4 at step 10, with width-spread $\leq 2.5\times$ within each variant), suggesting layer-wise normalization absorbs much of the width-dependence; naïve μP achieves 0.0% LR transfer under Adam. Each row has its own y-axis.

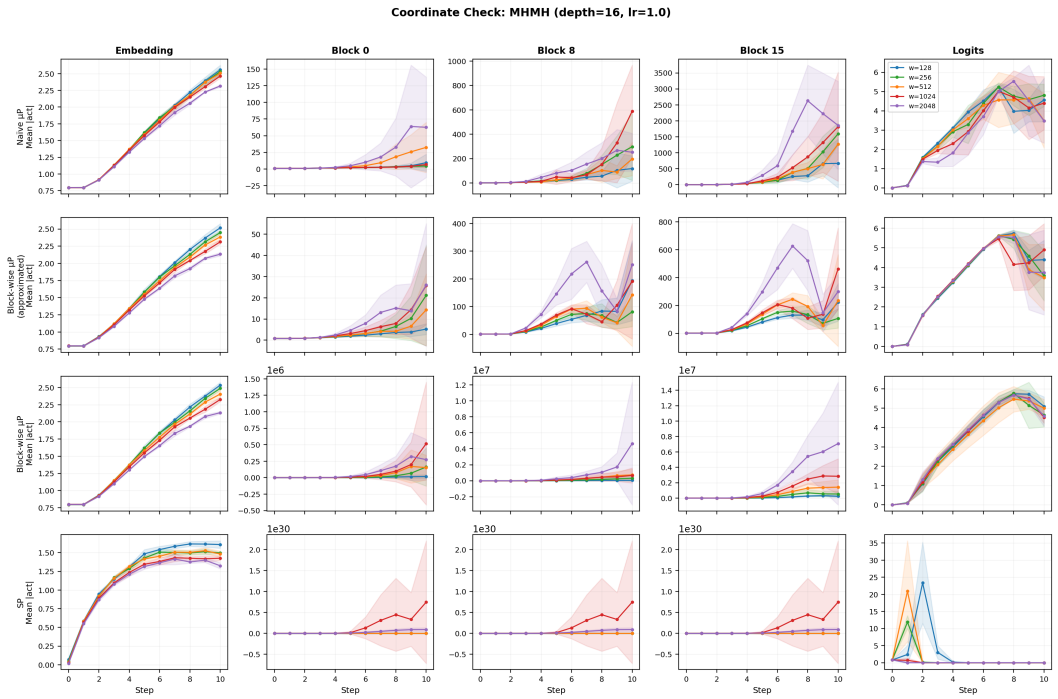


Figure 9: Coordinate check for MHMH (d=16, widths 128–2048). Same pattern as HHHH. Each row has its own y-axis.

Key observation: the activation magnitude is monotonic with depth (block_0 < block_8 < block_15), suggesting cumulative residual stream amplification.

Width-spread summary. Table 32 summarizes each coordinate check by one number. For each hidden block, we average the mean absolute activation over the five seeds at step 10 and take the ratio of its largest to its smallest value across the five widths; the table reports the median of this ratio over the 16 blocks. A width-invariant block gives 1. The pure Transformer (`TT`, Figure 10) and the GLA–Transformer hybrid (`GTGT`, Appendix C.5.3) are included for comparison.

Table 32: Median width-spread of the coordinate check (step 10, widths 128–2048, five seeds, d=16): for each hidden block, the largest over the smallest seed-averaged mean absolute activation across widths, and the median over the 16 blocks. A value of 1 means width-invariant.

Arch	Scheme	Median width-spread
`TT`	Naïve μP	1.52
	SP	819
`GTGT`	Naïve μP	1.23
	SP	7.3×10^{3}
`HHHH`	Naïve μP	2.78
	Block-wise (approx.)	2.45
	Block-wise (faithful)	5.71
	SP	6.0×10^{7}
`MHMH`	Naïve μP	4.23
	Block-wise (approx.)	3.33
	Block-wise (faithful)	65.67
	SP	1.4×10^{9}

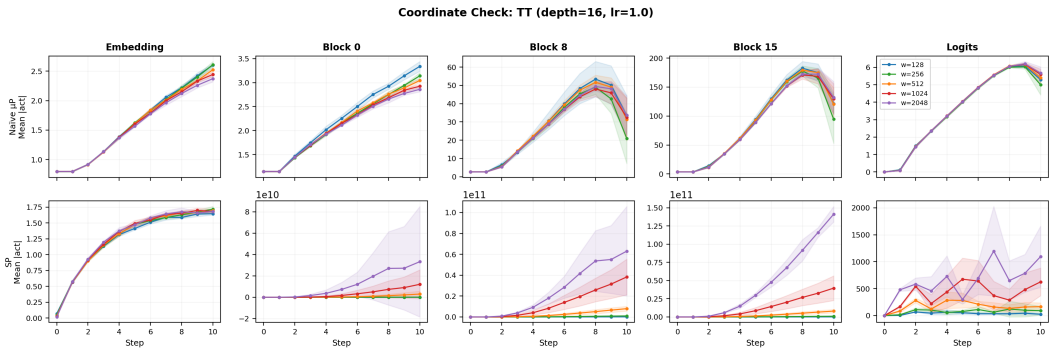


Figure 10: Coordinate check for TT (pure Transformer, d=16, widths 128–2048) with the same instrument. Row 1: naïve μP, width lines overlap (median block width-spread 1.52×, Table 32). Row 2: SP, activations diverge with width.

E.2 Gradient Measurements

On pure Transformers, Haas et al. [2025] identify a large-LR controlled-divergence regime where even SP achieves non-trivial feature learning at infinite width; our gradient measurements test whether AdamW's per-parameter normalization plays an analogous role across the SSM boundary in the heterogeneous hybrid setting. We measure per-component gradient RMS at initialization across widths 128–1024 to quantify the per-component gradient imbalance.

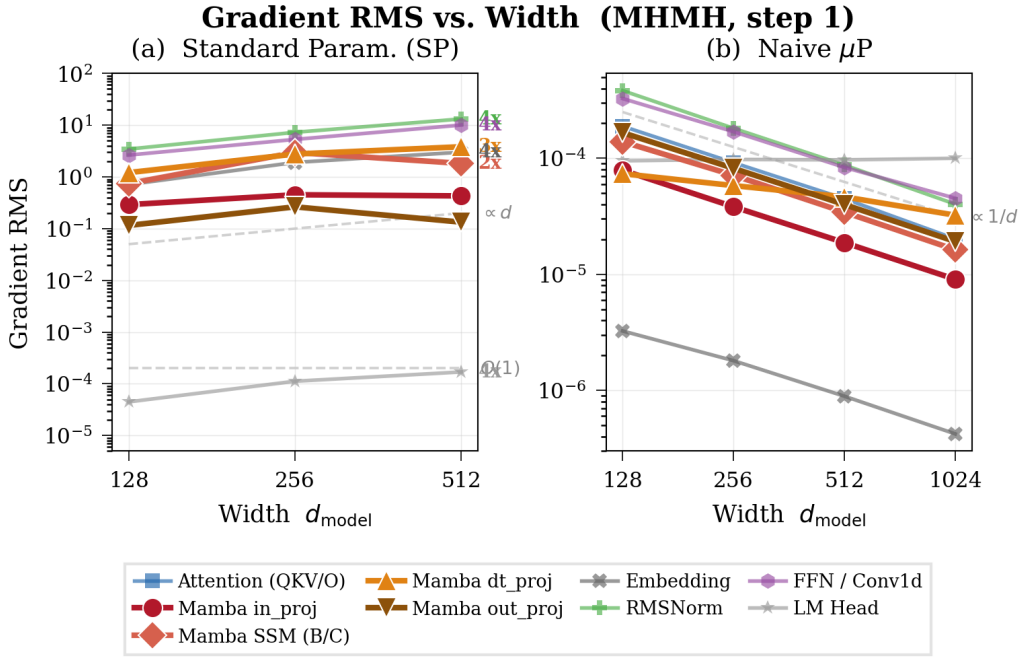


Figure 11: Gradient RMS per component vs. width for MHMH at step 1, under SP (widths 128–512) and naïve μP (128–1024). Components differ widely in gradient scale. Under Adam, the $1/\sqrt{v}$ denominator normalizes these disparate scales.

These measurements support the two-condition decomposition (Section 5): naïve μP satisfies Condition 1 (global update-to-weight ratio) but not per-component gradient balance. Adam's adaptive normalization restores Condition 2, enabling LR transfer despite the gradient imbalance.

Total gradient norm at initialization. At the aggregate level, the total gradient norm at initialization (single forward-backward pass on the MT interface, no optimizer step) shows the same SP-vs-μP separation seen at the per-component level (Table 33). SP produces gradient norms that grow from 0.6 to 2.5 across w=128–2048, while both μP variants stay between 0.1 and 0.5 at the same widths, $5\times$ lower at w=2048.

Table 33: Gradient norm at initialization across widths and schemes (MT interface).

	Width				
Scheme	128	256	512	1024	2048
SP	0.6	0.9	1.2	1.7	2.5
Naïve μP	0.1	0.2	0.3	0.4	0.5
Block-wise μP	0.1	0.2	0.3	0.4	0.5

SP's gradient norm grows $\sim 4\times$ from w=128 to w=2048 ($0.60 \to 2.48$); both μP variants grow by a similar factor ($\sim 4\times$) but from a much lower base ($0.12 \to 0.50$). Notably, naïve and block-wise μP (both variants) produce nearly identical gradient norms at initialization; the divergence between them emerges only during training under Adam (Section 5).

Joint (N_u, N_x, d) gradient-scaling sweep. The 1D measurement above varies only d_{model} at fixed `ssm_expand` $= 2$, so $N_u = 2d$ is perfectly collinear with d and the 1D fit against N_u alone cannot separate a dependence on d from a dependence on N_u. To break this degeneracy we run a second, larger sweep varying the three axes independently: $d_{\text{model}} \in \{128, 256, 512\}$, `ssm_expand` $\in \{1, 2, 4\}$ (so $N_u =$ `expand` $\cdot\, d$ spans $[128, 2048]$), `ssm_d_state` $\in \{8, 16, 32, 64\}$ (so N_x varies independently), each for 3 seeds and 200 training steps.

Mapping to Vankadara's A, B, C. The Mamba block parameters in our measurement correspond to the components of Chennuru Vankadara et al. [2024]'s framework as follows: `mamba_A_log` is the SSM state-transition matrix A; the B and C matrices in Vankadara's prescription are slices of the input-projection `x_proj` (the rows producing the per-step B and C at each position). We use `ssm_split_projections`=True for this sweep so that B and C are separate parameter groups, allowing Vankadara's per-component rules ($\eta_B \propto 1/\sqrt{N_u}$, $\eta_C \propto 1/(N_x\sqrt{N_u})$) to act at the granularity their derivation assumes; this matches the configuration used in the main width-transfer experiments (Table 1). The labels `mamba_in_proj` ($d_{\text{model}} \to 2d_{\text{inner}}$ input gate) and `mamba_out_proj` ($d_{\text{inner}} \to d_{\text{model}}$) are *not* B or C; they are the input/output projections that surround the SSM core.

Effective update at step 1: the direct μP test under SGD. μP theory constrains the *update* ($\eta_{\text{eff}}(W, \text{scheme}) \cdot \text{grad}(W)$), not the raw gradient. Vankadara's prescription predicts that the effective update is width-invariant ($\alpha(N_u) = 0$) under SGD when the per-component rules are properly applied. We compute the step-1 effective update RMS for $A_{\log}$, B, and C under both naïve μP and the faithful Vankadara prescription, and fit $\log u \sim \alpha \log N_u + \gamma \log d + \log C$ over the joint $(d,$ `expand`, `d_state`$)$ grid (3 seeds, $n = 108$ per row; the headline result is reported in the main paper as Table 6).

Vankadara's prediction holds at init for B and C but fails for $A_{\log}$. At step 1 the faithful prescription produces near-zero N_u dependence for B ($\alpha = +0.006$) and C ($\alpha = +0.029$): η_B and η_C cancel the underlying gradient growth and deliver the predicted width-invariant update. For $A_{\log}$, however, the faithful update has $\alpha = +1.00$, growing approximately linearly with N_u. This is not a finite-width residual: under naïve μP the $A_{\log}$ update already has $\alpha = +0.86$ at step 1, so $\eta_A \propto N_u$ is not canceling an underlying $1/N_u$ decay (no such decay is present in our gradients), but *amplifying* the existing N_u-dependence. This is consistent with the simplified-ZOH discretization gap (Appendix A.4): Vankadara's derivation assumes true ZOH, where $A_{\log}$'s gradient may decay as predicted, but our implementation uses the field-standard simplified form (Mamba library, Mamba-2 [Dao and Gu, 2024], Jamba [Jamba Team, 2025], Nemotron-H [NVIDIA, 2025], Falcon-H1 [Zuo et al., 2025]).

The init-level invariance does not survive training. Across the stable window (steps 100–199), the faithful $\alpha(N_u)$ for B drifts from $+0.006$ to $+1.35$ ($\Delta\alpha = +1.34$) and for C from $+0.029$ to

$+0.71$ ($\Delta\alpha = +0.68$); for $A_{\log}$, faithful drifts from $+1.00$ to $+3.42$ ($\Delta\alpha = +2.42$). The explicit per-component LR rules thus achieve their predicted invariance only at step 1 for B and C; during training the structure is destroyed at a rate *larger* for the faithful prescription than for naïve ($\Delta\alpha$ for naïve stays in $[0.33, 0.68]$ across all three components). The init-level Vankadara replication is therefore a 1-step phenomenon, not a property the framework continues to enforce as the optimizer trajectory leaves the infinite-width regime; the LR-operating-point caveat below sharpens this reading.

Stable window: LR-specific operating points, not scaling laws. The stable-window exponents in Table 6 are at base LR $\eta = 1$ under SGD and should be read as LR-specific equilibrium operating points rather than fundamental scaling laws. Cross-checking naïve μP SGD stable-window fits at base LR $\eta = 1$ and $\eta = 10$, the coefficients inflate 4–18$\times$ while residuals remain near-Gaussian in both cases; alternative functional forms (semi-log, log+linear envelope) produce the same coefficient inflation, so LR-dependence is not a log-log artifact. The residual-shape distinction is robust across LRs and describes a structural difference between optimizers: AdamW's $g/\sqrt{v}$ introduces coordinate-wise coupling that disrupts log-linear structure (heavy-tailed residuals, excess kurtosis $+5$ to $+19$), while SGD's unit update reflects pure weight-trajectory dynamics (near-Gaussian residuals). The Table 6 stable-window column should be read as confirming that the init-level invariance for B and C does not persist along the trajectory, not as a precise width-scaling exponent for any particular LR.

Outlier structure (AdamW, naïve μP). Pooled log-residuals under AdamW are heavy-tailed for every scheme (excess kurtosis $+5$ to $+19$, skewness $+1$ to $+3$). Top-1% |residual| events cluster in specific $(d, \texttt{expand})$ cells rather than uniformly (Fig. 12): for naïve μP, $\texttt{expand} = 1$ is uniformly quiet ($\sim 0\%$), while $\texttt{expand} \in \{2, 4\}$ concentrate outliers at the extreme d values. Per-run trajectories show these as isolated single-step gradient collapses: transient update-instability events, not Gaussian measurement noise. This clarifies that the low R^2 of AdamW stable-window fits reflects structured outliers (configuration-dependent transient spikes) rather than purely random scatter.

The cleanliness–performance inversion under AdamW. The same effective-update analysis run on AdamW data (not shown for length) inverts the ordering: at the stable-window operating point under AdamW, the faithful prescription's $A_{\log}$ update has the cleanest log-log structure (highest R^2), but block-wise underperforms naïve on hybrid training (Appendix A.4, Appendix E.3). This is consistent with the redundancy story (Section 5.3): under SGD, $\eta_A \propto N_u$ is doing useful but in our setting overcorrected work for $A_{\log}$, while under AdamW the $g/\sqrt{v}$ normalization already supplies width-stable per-component balance, so adding the explicit rule slows convergence without changing the convergent optimum. Clean stable-window gradient structure under AdamW is therefore a *diagnostic* that a per-parameter LR rule is firing, not a predictor of training success.

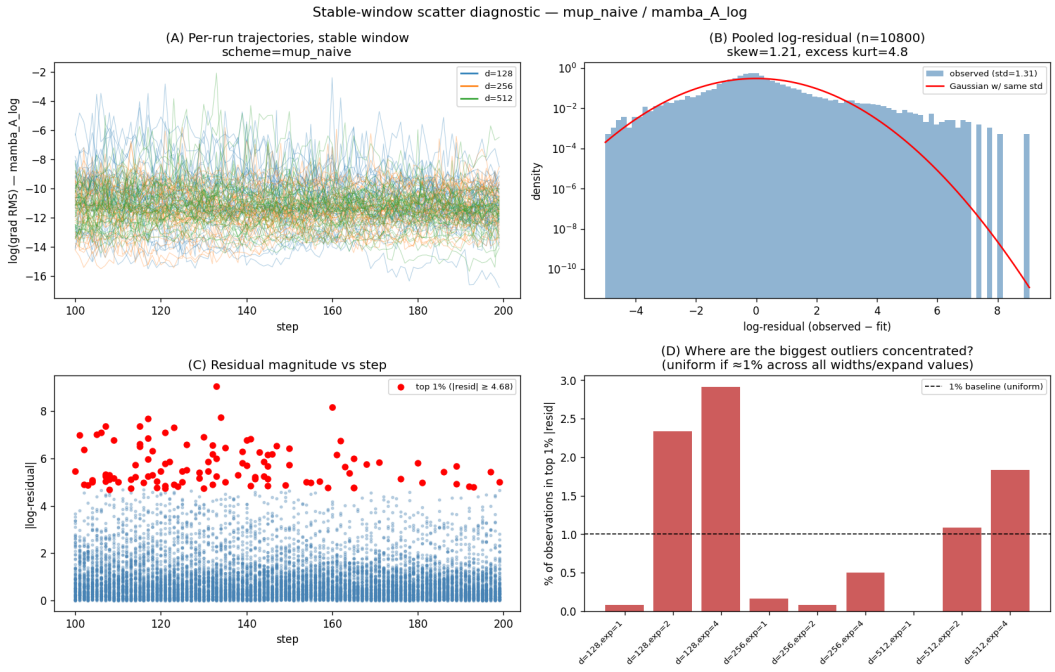


Figure 12: Stable-window scatter diagnostic for MHMH naïve μP, $A_{\log}$. (A) Per-run log-gradient trajectories over steps 100–199 show isolated single-step collapses (downward spikes) rather than smooth noise. (B) Pooled log-residual histogram against a Gaussian with matching standard deviation. The residuals are strongly heavy-tailed (excess kurtosis $+4.8$). (C) Top-1% $|\text{residual}|$ values scattered throughout the stable window; no concentration at early or late steps. (D) Outlier concentration by $(d, \texttt{expand})$ cell: non-uniform, concentrated in $\texttt{expand} = 2, 4$ cells; $\texttt{expand} = 1$ quiet. The outliers are transient update-instability events structured by configuration, not uniform measurement noise.

E.3 Optimizer Ablations

We ablate Adam's components one at a time while holding the parameterization fixed. Adam differs from SGD in three ways: adaptive LR ($g/\sqrt{v}$), momentum (β_1), and weight decay (λ).

E.3.1 SGD vs AdamW at 2k Steps

56 configs: HHHH/MHMH $\times$ $w \in \{256, 1024\}$ $\times$ SGD/AdamW $\times$ 7 LRs, 2k steps on WikiText-103.

SGD barely trains at 2k steps (loss $\sim$7–9 vs random baseline 11.5), while AdamW achieves loss $\sim$4.1–4.4 at η=1.0. Inconclusive for transfer comparison at this short training horizon.

E.3.2 SGD at 20k Steps with Extended LR Range

To ensure SGD trains meaningfully, we extend to 20k steps and LR up to 100.0. 48/48 configs complete (MHMH + HHHH, both widths, AdamW + SGD; Table 34).

Table 34: SGD vs AdamW for MHMH and HHHH (naïve μP, WIKITEXT-103, 20k steps, warmup 1000).

Arch	Optimizer × Width	η=0.3	η=1.0	η=3.0	η=10.0	η=30.0	η=100.0
MHMH	AdamW w=256	3.566	**3.518**	3.616	4.018	9.909	5.445
	AdamW w=1024	3.226	**3.129**	3.567	3.667	3.838	*10.5*
	SGD w=256	6.927	6.503	6.176	5.668	5.238	**5.121**
	SGD w=1024	7.296	6.736	6.329	6.147	5.687	**5.176**
HHHH	AdamW w=256	3.505	**3.487**	3.581	*div*	4.542	*10.9*
	AdamW w=1024	3.160	**3.082**	3.476	3.691	3.829	*10.9*
	SGD w=256	6.870	6.374	5.986	5.506	5.233	**5.122**
	SGD w=1024	7.389	6.856	6.399	5.841	5.403	**5.174**

SGD requires at least $100\times$ higher LR than Adam and is 46% worse at its best tested LR (MHMH w=256: SGD 5.121 vs AdamW 3.518). Critically, SGD shows *reversed width scaling*: w=1024 performs worse than w=256 at every LR up to η=100 (delta $+1.1\%$ to $+8.6\%$). Under AdamW, wider models perform better as expected, and η=1.0 remains optimal across widths (0.0% gap).

Above η=100. Every SGD row of Table 34 has its best loss at η=100, the top of the grid. We extend HHHH in the same configuration (the naïve-μP parameter groups of Section 3.2, 20k steps, one seed) to $\eta \in \{300, 1000, 3000, 7000\}$ (Table 35). The argmin is η=1000 at w=256 and η=3000 at w=1024, and the lowest LR that diverges is 3000 at w=256 and 7000 at w=1024; in this configuration both the argmin and the stability edge move with width. The extension is limited to HHHH: MHMH above η=100 was run only at w=256.

Table 35: HHHH under SGD above η=100, in the configuration of Table 34 (naïve-μP parameter groups, WIKITEXT-103, 20k steps, one seed): final training loss. The η=100 column repeats Table 34. Bold marks the argmin of each row; *div*: diverged; –: not run.

Optimizer × Width	η=100	η=300	η=1000	η=3000	η=7000
SGD w=256	5.122	5.077	**5.067**	*div*	–
SGD w=1024	5.174	5.033	5.010	**5.004**	*div*

E.3.3 Approximated Block-wise μP with SGD

24 additional configs: HHHH/MHMH $\times$ $w \in \{256, 1024\}$ $\times$ 6 LRs, approximated block-wise μP + SGD (momentum=0.9), 20k steps. This completes the 2×2 matrix of $\{\text{naive}, \text{approximated block-wise}\} \times \{\text{AdamW}, \text{SGD}\}$.

Table 36: 2×2 matrix: parameterization × optimizer (MHMH, best LR at each width).

Parameterization	Optimizer	Best LR @256	Loss @256	Best LR @1024	Loss @1024
Naïve μP	AdamW	1.0	3.518	1.0	**3.129**
Naïve μP	SGD	100.0	5.121	100.0	5.176
Approximated block-wise μP	SGD	100.0	5.122	100.0	5.408

Approximated block-wise μP does not fix SGD's reversed width scaling. Both SGD variants show w=1024 performing worse than w=256, with the gap *larger* under approximated block-wise μP for MHMH ($+5.6\%$ vs $+1.1\%$).

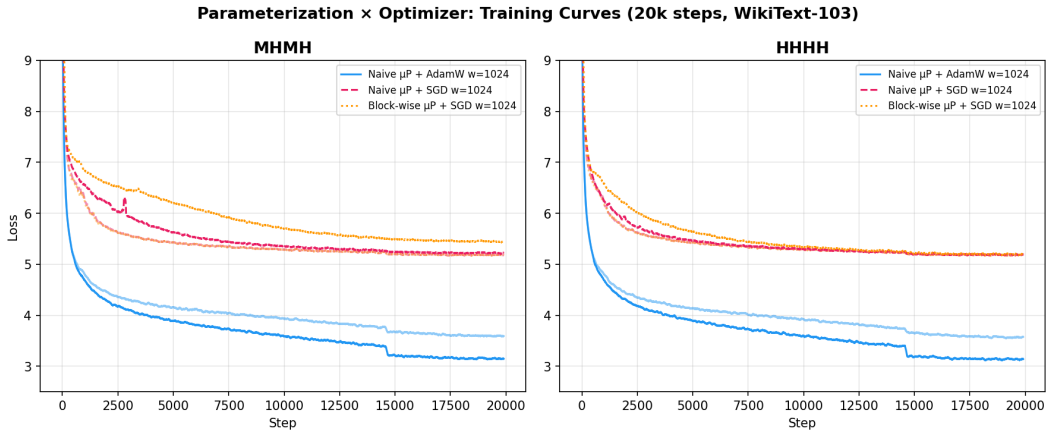


Figure 13: Training curves for three parameterization–optimizer combinations. AdamW (blue) separates by width (wider = better); SGD variants (pink, orange) overlap or show reversed scaling. Approximated block-wise μP does not improve SGD over naïve μP.

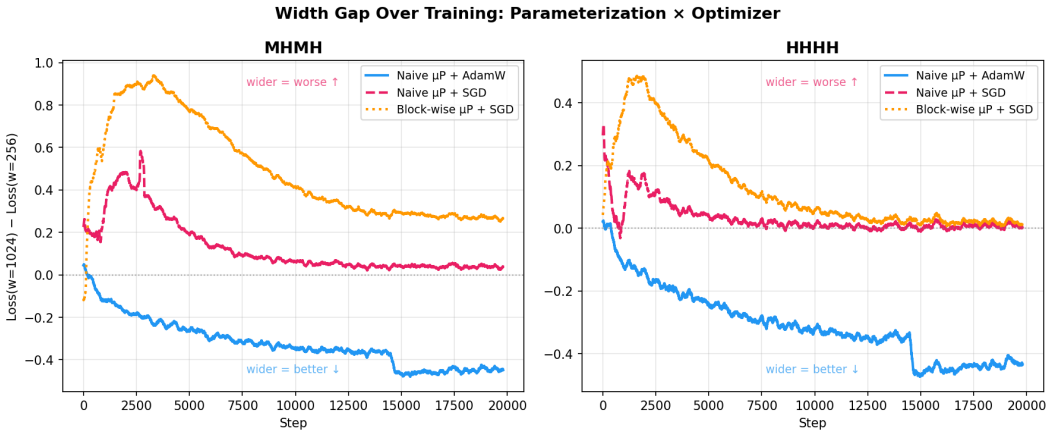


Figure 14: Width gap (loss at w=1024 minus loss at w=256) over training. Positive values indicate reversed scaling (wider = worse). AdamW (blue) achieves healthy scaling; both SGD variants (pink, orange) show persistent reversed scaling.

E.3.4 Adam without Weight Decay

36 configs: MM/HHHH/MHMH $\times$ $w \in \{256, 1024\}$ $\times$ WD $\in \{0.0, 0.1\}$ $\times$ 3 LRs (Table 37). η=1.0 transfers for all 3 architectures with WD=0.0; weight decay shifts the loss by at most 0.4% and leaves the optimal LR unchanged.[10] Weight decay is therefore not load-bearing for the LR-transfer property in our setting.

[10] At w=1024, `MM` with weight decay reaches a 0.1% lower loss at η=3.0 than at η=1.0, within the flat basin around the optimum. The ablation runs for 2k steps, shorter than the weight-decay timescale at w=1024.

Table 37: Adam no-WD ablation (WIKITEXT-103, 2k steps, naïve μP).

	η=0.3	η=1.0	η=3.0
HHHH w=256 WD=0	4.573	**4.411**	4.517
HHHH w=256 WD=0.1	4.561	**4.404**	5.013
HHHH w=1024 WD=0	4.392	**4.090**	4.471
HHHH w=1024 WD=0.1	4.381	**4.102**	4.161
MHMH w=256 WD=0	4.411	**4.234**	4.296
MHMH w=256 WD=0.1	4.411	**4.217**	4.234
MHMH w=1024 WD=0	4.455	**4.165**	4.300
MHMH w=1024 WD=0.1	4.451	**4.170**	4.254
MM w=256 WD=0	4.557	**4.355**	4.479
MM w=256 WD=0.1	4.549	**4.361**	4.447
MM w=1024 WD=0	4.599	**4.397**	4.452
MM w=1024 WD=0.1	4.593	4.402	**4.397**

E.3.5 Momentum Ablation

56 configs: HHHH/MHMH $\times$ $w \in \{256, 1024\}$ $\times$ 7 LRs $\times$ {Adam (β_1=0), SGD (no momentum)} (Table 38).

Table 38: Optimizer ablation for `MHMH` and `HHHH` (2k steps, WIKITEXT-103, naïve μP). Final loss at each LR; bold marks the optimum within each row.

Arch	Optim $\times$ Width	η=0.001	η=0.003	η=0.01	η=0.03	η=0.1	η=0.3	η=1.0
MHMH	AdamW w=256	9.968	8.215	6.638	5.625	4.898	4.412	**4.219**
	AdamW w=1024	9.303	7.950	6.665	5.706	5.020	4.450	**4.160**
	Adam (β_1=0) w=256	9.951	8.194	6.726	5.653	4.954	**4.448**	4.511
	Adam (β_1=0) w=1024	9.289	7.935	6.718	5.737	5.059	4.608	**4.418**
	SGD w=256	*11.5*	*11.5*	*11.5*	*11.4*	9.999	8.462	**7.626**
	SGD w=1024	*11.5*	*11.5*	*11.5*	*11.4*	*10.9*	8.821	**7.911**
	SGD (no momentum) w=256	*11.5*	*11.5*	*11.5*	*11.5*	*11.5*	*11.3*	**9.820**
	SGD (no momentum) w=1024	*11.5*	*11.5*	*11.5*	*11.5*	*11.5*	*11.4*	*10.9*
HHHH	AdamW w=256	9.974	8.249	6.729	5.742	5.051	4.561	**4.392**
	AdamW w=1024	9.277	7.923	6.719	5.694	4.961	4.386	**4.099**
	Adam (β_1=0) w=256	9.957	8.230	6.778	5.767	5.104	4.615	**4.470**
	Adam (β_1=0) w=1024	9.263	7.909	6.719	5.720	5.020	4.492	**4.465**
	SGD w=256	*11.5*	*11.5*	*11.5*	*11.3*	9.298	8.415	**7.732**
	SGD w=1024	*11.5*	*11.5*	*11.5*	*11.4*	*10.9*	8.753	**7.955**
	SGD (no momentum) w=256	*11.5*	*11.5*	*11.5*	*11.5*	*11.5*	*11.3*	**9.398**
	SGD (no momentum) w=1024	*11.5*	*11.5*	*11.5*	*11.5*	*11.5*	*11.4*	*10.8*

Removing Adam's momentum costs 2–9% in loss (e.g., MHMH w=1024: $4.160 \to 4.418$, $+6.2\%$; HHHH w=1024: $4.099 \to 4.465$, $+8.9\%$) but leaves transfer intact: $3/4$ (arch, width) cells keep η^*=1.0, with one borderline shift to η=0.3 (Adam β_1=0 on MHMH w=256). Removing SGD's momentum is catastrophic: loss climbs to 9.4–10.9, barely below the random baseline of 11.5, an additional 22–38% degradation that wipes out training entirely. Momentum therefore aids optimization speed but is not the transfer mechanism; only Adam's $g/\sqrt{v}$ rescaling is.

Note on the LR grid boundary. The grid in Table 38 stops at η=1.0, so any AdamW or Adam (β_1=0) optimum reported at η=1.0 is at the upper boundary of *this* grid. The width-transfer sweep (Table 1) extends to η=3.0 on the same (arch, width) cells under naïve μP + AdamW and confirms η^*=1.0 remains optimal there, so the AdamW optimum is genuinely η=1.0 rather than an artifact of the local boundary. The SGD rows here also peg the boundary, but the wider sweep in Table 34 extends to η=100 and, with Table 35, shows SGD's best loss stays near 5 at both widths, never approaching AdamW's $\sim$ 3.1; the SGD-vs-AdamW comparison is therefore qualitative and unchanged by the boundary.

E.3.6 Adam Epsilon Sensitivity

We test whether the LR transfer claim is sensitive to AdamW's ϵ. We sweep $\epsilon \in \{10^{-10}, 10^{-8}, 10^{-6}, 10^{-4}\}$ on MHMH at $w \in \{256, 1024\}$ and $\eta \in \{0.5, 1.0, 1.5\}$ for 2k steps on WIKITEXT-103 (24 configs, Table 39).

Table 39: AdamW ϵ sensitivity for MHMH (WIKITEXT-103, 2k steps, naïve μP). Each η block has paired rows for w=256 and w=1024 so width-invariance can be read directly across the row pair; bold marks the optimum across the four ϵ values within each row.

η	w	ϵ=10^{-10}	ϵ=10^{-8}	ϵ=10^{-6}	ϵ=10^{-4}
0.5	256	4.306	4.293	**4.260**	4.557
	1024	4.287	4.286	**4.281**	4.852
1.0	256	4.236	4.209	**4.177**	4.316
	1024	4.176	4.167	**4.150**	4.447
1.5	256	4.219	4.211	**4.163**	4.243
	1024	4.156	**4.148**	4.209	4.302

At the μP optimum η=1.0, varying ϵ over $\{10^{-10}, 10^{-8}, 10^{-6}\}$ shifts the loss by less than 0.07 nats at both widths, and η=1.0 stays within 0.5% of the best LR in every cell; ϵ=10^{-6} slightly outperforms the default 10^{-8} ($\sim$ 0.03 nats better at both widths). At ϵ=10^{-4} the loss degrades sharply (+0.10 nats at w=256, +0.30 nats at w=1024): a large ϵ suppresses Adam's per-parameter normalization. This is consistent with the two-condition decomposition (Section 5.2): when ϵ is small relative to typical $\sqrt{v_W}$, the $g/\sqrt{v}$ normalization operates as designed and Condition 2 holds; when ϵ becomes comparable to $\sqrt{v_W}$, AdamW degenerates toward SGD-like behavior and Condition 2 fails, producing the same width-dependent degradation we observe under SGD itself.

E.3.7 Sophia and Muon

We run two further optimizers on HHHH and MHMH with naïve μP at $w \in \{256, 512, 1024\}$, 20k steps on WIKITEXT-103, three seeds each, and report validation loss. Sophia [Liu et al., 2024] (SophiaG, β=(0.965, 0.99), ρ=0.04, Hessian estimate every 10 steps) uses the naïve-μP parameter groups. Muon [Jordan et al., 2024] (momentum 0.95) orthogonalizes the updates of the hidden weight matrices; that group uses Muon's own update scaling at η_{base}, without the μP multiplier, and the remaining parameters (embeddings, output head, norms, SSM parameters) use AdamW with the naïve-μP groups. Sophia keeps its optimum at every width, while Muon's optimum moves at w=1024 (Table 40). Sophia's grid has three points spaced about 3$\times$ apart, so its 0.00% gap holds at that resolution. A first run of the same grids without a held-out evaluator selected the same η^* as the rerun at w=256 and w=1024 for every seed, with Muon training-loss gaps of 3.3–11.3%; Table 40 reports the rerun that adds the evaluator. Sophia and Muon differ in prescription as well as in normalizer, so the comparison does not isolate per-element normalization.

Table 40: Sophia and Muon on WIKITEXT-103 (20k steps, three seeds, validation loss). η^* is the argmin of the seed-mean loss; the gap is the transfer gap from w=256 to w=1024, for the seed mean and per seed. LR grids: Sophia $\{0.03, 0.1, 0.3\}$, Muon $\{0.002, 0.005, 0.01, 0.02, 0.05, 0.1\}$.

Optimizer	Arch	η^* w=256	w=512	w=1024	Gap (mean)	Gap (per seed)
Sophia	HHHH	0.1	0.1	0.1	0.00%	0.00% (3/3)
Sophia	MHMH	0.1	0.1	0.1	0.00%	0.00% (3/3)
Muon	HHHH	0.01	0.01	0.005	3.58%	2.29–5.74%
Muon	MHMH	0.01	0.01	0.005	7.83%	5.62–10.38%

E.3.8 Confound Resolution Summary

Table 41: Adam-component contribution to LR transfer: which feature is required vs. incidental.

Adam component	Role for LR transfer	Evidence
Adaptive LR ($g/\sqrt{v}$)	**Load-bearing**	Only Adam variants transfer
Momentum (β_1)	Not load-bearing	2–9% degradation, transfer preserved
Weight decay (λ)	Not load-bearing	$\leq$0.4% effect

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We provided the statement of our contribution in the abstract and in the introduction shown with the list. We also provided our scope in both sections as the hybrid architectures in the practical setting.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discussed the limitations of our work in the paper, particularly in the context of limitation of practical scale we tested and the assumptions made on the discretization that differs from previous work.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper does not contain new theorems or formal proofs. We propose a two-condition decomposition of LR transfer that draws on existing theory (Yang et al. 2021 and Yang and Littwin 2023 for Condition 1; Tomihari and Sato 2026 for Condition 2) and use the two conditions as explanatory diagnostics, examined empirically through optimizer ablations and a per-component scaling test. Whether they are jointly sufficient at infinite width is left open in Section 6.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: We provided the details of our experimental setup in the paper, including the architecture composition, parameterization schemes, training details, and evaluation metrics. We also provided the full experiment matrix in the appendix.

Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: The code is released under the MIT license at `https://github.com/argyroneta-aquatica/hybrid-mup`: the model, the parameterization schemes, a SLURM script with the exact commands for the experiments behind each table and figure, the recipes of the container images they ran in, and scripts that download and prepare every dataset (all public). The per-run result files and the analysis and plotting scripts are not released; per-run summary tables are included instead.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We provided the details of our experimental setup throughout the paper and the appendix. This includes the architectural details of blocks, the parameterization schemes, and the training details.

   Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: Our experiments required a large amount of compute for multiple runs, and we were not able to report the error bars for all of our experiments. We reported the error bars for the multi-seed validation runs in the appendix, and it provided with how it were calculated and which seed it used.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Section 3 reports total project compute (∼5,500 GPU-hours across 4,500+ completed configurations); Appendix A.1 details the GPU mix used (NVIDIA A6000 48 GB, A5000 24 GB, and A100 80 GB on an internal SLURM cluster) and the per-architecture tuning-cost saving; Appendix D contains the full configuration matrix. The total counts every cluster job of the project, including preliminary and failed runs that are not reported in the paper, such as early runs with an η_A scaling bug.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We conducted our research to investigate the learning rate transfer of hybrid architectures. Our research does not involve societal impacts.

Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: Our work is focused on the zero-shot learning rate transfer of hybrid architectures. Our work is not directly related to potential societal impacts, except for the efficient training of large models.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: Our work focuses on the analysis of the learning rate transfer in hybrid architectures, and we do not release any new model or dataset.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All third-party assets are credited in the paper or listed here, with their licenses: WIKITEXT-2/WIKITEXT-103 [Merity et al., 2017] (CC BY-SA 3.0) and FINEWEB-EDU [Penedo et al., 2024] (ODC-BY 1.0) for datasets; the Mamba reference implementation [Gu and Dao, 2024] (Apache 2.0) for the SSM block code path; the Pythia [Biderman et al., 2023] (Apache 2.0) and Mamba baseline checkpoints (Apache 2.0) for downstream-evaluation comparisons; causal-conv1d (BSD-3-Clause) for the Mamba convolution; flash-linear-attention (MIT) for the GLA blocks; tiktoken (MIT) for tokenization; GPT-NeoX (Apache 2.0) for training the baselines; the official Sophia and Muon implementations (MIT); and lm-evaluation-harness (MIT) for zero-shot evaluation on HellaSwag, PIQA, WinoGrande (CC BY), ARC (CC BY-SA 4.0), and LAMBADA, each cited at first use.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: We release our code (Question 5).

    Guidelines:

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

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

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

Answer: [N/A]

Justification: Our work does not include experiments with human subjects or crowdsourcing.

Guidelines:

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

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

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

Answer: [N/A]

Justification: Our work does not include research with human subjects.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: Language models are our main object of the study, rather than the methodological component of our research. Our research focuses on the learning rate transfer of the hybrid architectures.

Guidelines:

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