ELASTIC SELECTIVE SPECTRAL HYBRIDS FOR TRAIN-ONCE, EXPORT-MANY BUDGETED INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Deploying a language model under different computing and latency budgets calls for compact models with different quality–cost trade-offs. Towards this end, elastic spectral state space models provide ordered and temporally decomposed channels that can be truncated, but they are associated with linear time-invariant filters that cannot selectively preserve relevant past information or forget irrelevant information as the context evolves. To address this, we introduce the Elastic Selective Spectral Hybrid (ESSH), which realizes each Hankel spectral channel as an independent recurrent unit using fitted damped rotation modes. It also features an input-dependent decay and write/read gates that make temporal retention and state update input-dependent while preserving channel-wise truncation and a structured recurrence for efficient execution. ESSH combines these selective spectral mixers with sliding-window attention and jointly trains multiple capacities by reducing spectral-channel count and feed-forward width at different rates through a two-rate capacity map with full-model distillation. The resulting models support chunked parallel training and fused recurrent decoding while avoiding computation for discarded channels. At full capacity, ESSH achieves language-modeling quality comparable to similarly sized independently trained models, while smaller exports exhibit a smooth quality–cost trade-off. We validate the effectiveness of the proposed framework using language understanding, retrieval, cross-domain text, and DNA experiments, by assessing quality retention and the trade-off against independently trained and elastic baselines. At the 1.53B model configuration, fused batch-one decoding takes 1.37 ms per token on a B300, providing a 2.14–$2.80\times$ speedup over the tested Mamba-2 and Mamba-3 implementations and $3.03\times$ over Transformer++ at matched parameter counts.

1 INTRODUCTION

Language models are deployed under different computing and latency limits. A model that fits one deployment setting may be too expensive in another. This motivates the need for models with different sizes that can operate under different deployment budgets. Training an independent model for every deployment budget is costly. One approach is to start from a pretrained model and obtain smaller models through post-training compression or adaptation (Muralidharan et al., 2024; Taghibakhshi et al., 2025b). However, different deployment sizes may require additional size-specific pruning, adaptation, or distillation. Instead, elastic models jointly train nested subnetworks of different capacities within a single training procedure (Yu et al., 2019; Cai et al., 2020).

A key design choice in elastic models is how model capacity should be reduced as the deployment budget decreases. Existing methods define capacity through different internal dimensions. MatFormer varies the width of feed-forward blocks (Devvrit et al., 2024), while MatMamba varies the inner width and corresponding number of heads within the recurrent mixer (Shukla et al., 2024). A complementary approach is to define capacity through the temporal basis used for sequence mixing (Voelker et al., 2019). Compared with reducing generic internal dimensions, this directly controls the temporal components used to represent the input history. For example, Hankel spectral models provide an ordered spectral basis, making the number of retained channels a structured axis for elastic recurrent models (Hazan et al., 2017; Agarwal et al., 2023).

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

Existing elastic spectral state space models (ES–SSM) exploit this ordering by varying the number of retained spectral channels across deployment budgets (Song et al., 2026). However, their temporal filters remain linear time-invariant (LTI), which limits their ability to adapt how information is retained over time to the input content. In particular, the resulting temporal dynamics cannot selectively preserve relevant past information or forget irrelevant information as the context evolves. We therefore seek a new selective spectral recurrence that preserves the elastic spectral structure while making state update and retention input-dependent and keeping the recurrence efficient.

To do so, we represent each Hankel channel using damped rotation modes and introduce input-dependent mechanisms for controlling state retention. The resulting selective spectral mixer preserves independently truncatable channels and supports efficient parallel training and recurrent decoding. To further enhance model capability, we incorporate these mixers into a hybrid architecture combining state-space modeling with attention, and jointly train multiple capacities by varying both spectral-channel count and feed-forward width. Our contributions are:

- **Truncatable selective spectral channels.** We realize each Hankel spectral channel as an independent recurrent unit built from fitted damped rotation modes, with input-dependent decay and write/read gates. This allows state updates to be input-dependent while preserving whole-channel truncation and the structured transition for efficient execution.
- **Two-rate elastic training: spectral + feed-forward.** We jointly vary spectral-channel count and feed-forward width at different rates through a two-rate capacity map and train the resulting models with full-model distillation. This encourages compact subnetworks to maintain model quality across deployment budgets.
- **Efficient parallel training and recurrent decoding.** To improve computational efficiency, we derive a chunked parallel scan for the selective recurrence and implement GPU kernels for forward and backward passes. During decoding, we further use a fused recurrent path that combines the projections, state updates, and readouts, thereby avoiding computation for discarded channels.
- **Evaluation across quality, elasticity, and deployment cost.** We evaluate full-model quality, performance across capacities, and deployment cost against independently trained and elastic baselines. Across language modeling, retrieval, longer-context, cross-domain, and DNA experiments, targeted ablations further examine the effects of initialization, selective dynamics, elastic training, and the implementation.

2 Related Work

Train-once model families. Slimmable Networks and Once-for-All share parameters across subnetworks with different widths, depths, or resolutions, avoiding a separate training run for every deployment size (Yu et al., 2019; Yu & Huang, 2019; Cai et al., 2020). However, a smaller subnetwork retains only part of the shared computation, so the retained parameters must remain effective when evaluated independently. Nested dropout and Matryoshka representation learning address this by encouraging leading feature dimensions to carry useful task information (Rippel et al., 2014; Kusupati et al., 2022). MatFormer applies this idea to feed-forward width, while MatMamba applies it to the inner dimensions of Mamba-2 blocks (Devvrit et al., 2024; Shukla et al., 2024). Matryoshka mixture-of-experts extends nesting to expert sets, and Flextron introduces elastic choices into a pretrained Transformer (Wang et al., 2025; Cai et al., 2024). These works establish the general train-once principle: choose an elastic axis and train the retained subnetworks to remain useful. ESSH follows this principle, but introduces an ordered temporal axis through spectral channels and couples it with a separately controlled feed-forward axis.

Selective and spectral state space models. Mamba makes recurrent state update input-dependent, while Mamba-2 connects selective state-space computation to structured attention (Gu & Dao, 2024; Dao & Gu, 2024). Mamba-3 further develops the discretization and complex-valued dynamics (Lahoti et al., 2026). Gated DeltaNet combines gating with delta-rule state updates (Yang et al., 2025). These models provide selective state update and efficient recurrent computation, but not an ordered axis for reducing recurrent capacity. Hankel spectral methods provide such an order by approximating linear dynamical systems with an ordered set of temporal filters (Hazan et al., 2017; Agarwal et al., 2023; Liu et al., 2025). Elastic spectral SSMs use this ordering to vary channel count across deployment budgets (Song et al., 2026), but their temporal filters remain linear time-invariant and

cannot adapt state retention to the input content. ESSH combines the two complementary properties: Hankel-initialized channels provide an ordered axis for recurrent-capacity truncation, while input-dependent clocks and write/read gates make temporal retention selective.

Elastic hybrid models. Hybrid models such as Jamba and Zamba combine recurrent or SSM layers with attention, using recurrent computation for efficient sequence modeling and attention for stronger token interactions and retrieval (Lenz et al., 2025; Glorioso et al., 2024). Making such models elastic is harder because recurrent state, feed-forward layers, and attention contribute differently to model size and deployment cost. Minitron-SSM prunes SSM groups together with feed-forward width, embedding width, and depth, followed by distillation (Taghibakhshi et al., 2025b). Nemotron Elastic trains nested models within a pretrained model by jointly reducing SSM and feed-forward dimensions with multi-budget distillation (Taghibakhshi et al., 2025a). Star Elastic extends elastic training to hybrid mixture-of-experts models and separate budgets for thinking and answering (Taghibakhshi et al., 2026). These approaches introduce or refine elasticity after obtaining a pretrained parent model. In contrast, ESSH incorporates multiple deployment capacities directly into pretraining by varying both spectral-channel count and feed-forward width.

3 PRELIMINARIES

A linear state space model maps inputs $u(t) \in \mathbb{R}^{d_{\text{in}}}$ to outputs $y(t) \in \mathbb{R}^{d_{\text{out}}}$ through

$$h(t) = Ah(t-1) + Bu(t), \qquad y(t) = Ch(t) + Du(t), \tag{1}$$

where $h(t) \in \mathbb{R}^N$, $A \in \mathbb{R}^{N\times N}$, $B \in \mathbb{R}^{N\times d_{\text{in}}}$, $C \in \mathbb{R}^{d_{\text{out}}\times N}$, and $D \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$. With $h(0) = 0$, its temporal kernel is $G(\tau) = CA^\tau B$ and $y(t) = Du(t) + \sum_{\tau=0}^{t-1} G(\tau)u(t-\tau)$.

For symmetric A with eigenvalues in $[0, 1)$, Hankel spectral filtering represents the resulting temporal kernels using an ordered basis of temporal filters. Specifically, define the weighted geometric sequence $\mu(\lambda) = (1-\lambda)[1, \lambda, \ldots, \lambda^{L-1}]^\top$ and its second-moment matrix $Z = \int_0^1 \mu(\lambda)\mu(\lambda)^\top \, \mathrm{d}\lambda$. Let (σ_k, ϕ_k) be the eigenpairs of Z, ordered by decreasing σ_k, with $\|\phi_k\|_2 = 1$. The first K eigenvectors minimize mean squared projection error for this weighted single-pole family, which yields

$$G(\tau) \approx \textstyle\sum_{k=1}^K M_k \phi_k(\tau), \qquad \hat{y}(t) = \textstyle\sum_{k=1}^K M_k(\Phi_k * u)(t), \tag{2}$$

where $M_k \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ and Φ_k is the causal filter with samples $\phi_k(\tau)$. Derivations of this approximation are given in Appendix B.3. The direct term $Du(t)$ is kept separately.

Each term $M_k(\Phi_k * u)(t)$ defines one spectral channel. The ordering of the eigenpairs therefore provides a natural truncation axis: a full model uses $\bar{K}$ spectral channels, while a smaller model with $K < \bar{K}$ retains only the first K channels and discards channels $k > K$. However, the spectral ordering alone does not guarantee that the important information is aggregated in the low-index channels (after truncation). Thus, obtaining accurate smaller models requires explicitly training the retained channel sets to maintain task performance.

Elastic spectral SSMs (Song et al., 2026) build on this truncation rule by jointly training models with different retained channel counts. They additionally introduce a tokenwise two-layer GELU MLP to produce gate logits $s(t) = \mathrm{GateMLP}(u(t))$. With K retained channels, their output is

$$\hat{y}^{(K)}(t) = \textstyle\sum_{k=1}^K g_k^{(K)}(t) M_k(\Phi_k * u)(t), \qquad g_k^{(K)}(t) = \frac{1}{\sqrt{K}} \,\mathrm{sigmoid}(s_k(t)). \tag{3}$$

The gate changes the contribution of a fixed filtered feature, but the filter still depends only on lag τ. In the following, we introduce a selective mixer that preserves channel-wise truncation structure while making the temporal dynamics within each channel input-dependent.

4 METHOD

The proposed method aims to address the following challenges: (i) make the temporal dynamics input-dependent within spectral channels, (ii) maintain model performance across reduced-capacity tiers, and (iii) enable efficient parallel training and recurrent decoding. We refer to each model configuration with a specified spectral-channel count and feed-forward width as a capacity tier.

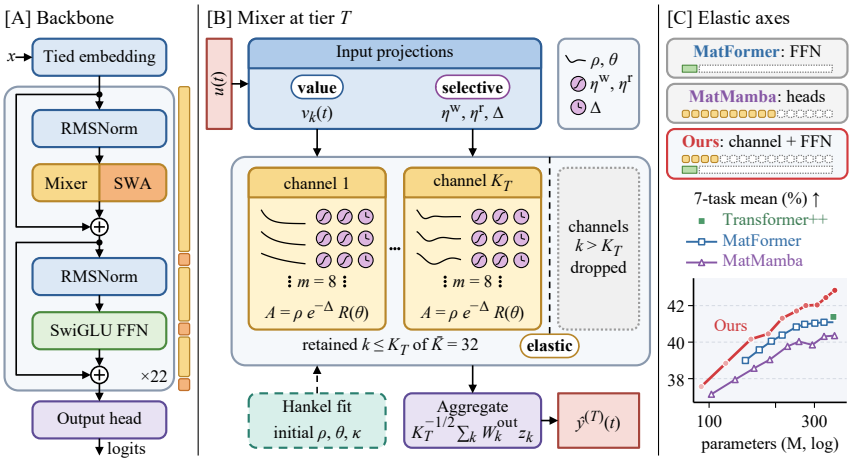

Figure 1: Overview of ESSH and its elastic axes. (A) The hybrid backbone combines spectral mixers with three sliding-window attention layers. (B) At tier T, the input produces channel values and token-dependent clock and write/read controls. Each retained spectral channel updates its m recurrent modes; channels $k > K_T$ are removed before the outputs are aggregated. (C) MatFormer varies feed-forward width, MatMamba varies mixer inner width, while ESSH varies both spectral-channel count and feed-forward width. The inset shows the quality–cost trade-off across capacities.

Towards this end, we first realize each spectral channel as an independent recurrent unit with input-dependent state updates (Sec. 4.1). We then combine channel truncation with feed-forward width reduction and jointly train the resulting capacity tiers with distillation from the full-capacity tier (Sec. 4.2). Finally, we use the same structured recurrence for a chunked parallel scan during training and a fused recurrent path during decoding (Sec. 4.3).

Figure 1 represents the overall hybrid architecture: Selective spectral mixers summarize inputs in recurrent states, while three sliding-window attention (SWA) layers provide direct access to recent token representations (Beltagy et al., 2020; Jiang et al., 2023). We vary capacity only inside the spectral mixers and feed-forward layers, and keep the residual width d and attention dimensions fixed across capacity tiers. The sliding-window configuration is specified in Appendix E.2.

4.1 Selective Spectral Mixer

The output gate in (3) rescales a fixed filtered feature, but cannot change how an earlier input propagates through intervening positions. To control this propagation within individual spectral channels, we approximate each channel using an independent recurrent realization of its fixed filter through damped rotation modes. For each channel, a matrix-pencil fit (Hua & Sarkar, 1990) approximates the Hankel filter with m damped rotation modes:

$$\phi_k(\tau) \approx \sum_{i=1}^{m} \rho_{k,i}^{\tau} \left(\kappa_{k,i}^{c} \cos(\theta_{k,i}\tau) - \kappa_{k,i}^{s} \sin(\theta_{k,i}\tau) \right). \tag{4}$$

Each damped rotation mode admits a two-dimensional recurrent realization, so this decomposition provides a compact recurrent realization of the fixed spectral filter. Here, $\rho_{k,i} \in (0,1)$ controls the intrinsic decay of the oscillator, $\theta_{k,i}$ is the rotation frequency, and $\kappa_{k,i}^{c}, \kappa_{k,i}^{s}$ are mode readout coefficients. The fitted parameters initialize the recurrence and remain trainable (with details in Appendix C.1). This recurrent realization approximates the temporal structure of the fixed spectral filter, but its dynamics are still independent of the input content.

To establish input dependence, output gating alone is not sufficient: it can suppress what is read at the current position, but cannot change what is propagated to future positions. We therefore make the state transition input-dependent through its decay and introduce separate write and read gates for input injection and readout. Let $u(t) \in \mathbb{R}^d$ be the normalized mixer input. Each channel forms a value $v_k(t) \in \mathbb{R}^P$, shared by its m modes, using the projection and channel-local normalization blend (details in Appendix C.2). Separate input projections produce clock increments $\Delta_{k,i}(t) > 0$ and write/read gates $\eta_{k,i}^{w}(t), \eta_{k,i}^{r}(t) \in (0,2)$. With mode state $H_{k,i}(t) \in \mathbb{R}^{2\times P}$, $e_1 = (1,0)^\top$, and

$\kappa_{k,i} = (\kappa^{\mathrm{c}}_{k,i}, -\kappa^{\mathrm{s}}_{k,i})^\top$, each mode evolves as

$$\begin{aligned} A_{k,i}(t) &= \rho_{k,i} \mathrm{e}^{-\Delta_{k,i}(t)} R(\theta_{k,i}), \qquad H_{k,i}(0) = 0, \\ H_{k,i}(t) &= A_{k,i}(t) H_{k,i}(t-1) + \eta^{\mathrm{w}}_{k,i}(t) e_1 v_k(t)^\top. \end{aligned} \tag{5}$$

The planar rotation is $R(\theta) = \left[\begin{smallmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{smallmatrix}\right]$. The clock $\Delta_{k,i}(t) > 0$ controls how much of the previous mode state is retained, the write gate controls how much of the current channel value enters the mode state, and the read gate controls how strongly that state contributes to the channel output. Summing $\eta^{\mathrm{r}}_{k,i}(t) H_{k,i}(t)^\top \kappa_{k,i}$ over the m modes gives the channel readout. A zero-lag correction (Appendix C.3) then gives the channel output $z_k(t)$. Because rotation preserves the state norm, the clock acts through the decay factor $\rho_{k,i} \mathrm{e}^{-\Delta_{k,i}(t)} < 1$. Larger clock increments suppress the previous mode state more strongly, while this factor approaches the intrinsic decay $\rho_{k,i}$ as the increment approaches zero.

A key design goal is to make the decay input-dependent while keeping the learned rotation frequency shared across positions. Because scalar decay and planar rotation commute, propagation from position s to t retains the closed form

$$\prod_{\tau=s+1}^{t} A_{k,i}(\tau) = \rho_{k,i}^{t-s} \mathrm{e}^{-\sum_{\tau=s+1}^{t} \Delta_{k,i}(\tau)} R\big(\theta_{k,i}(t-s)\big). \tag{6}$$

For $s = t$, the product is defined as the identity matrix. In (6), intervening tokens modulate state retention while rotation depends only on token lag, preserving the structured transition product used for parallel training in Sec. 4.3.

Furthermore, because each channel computes its own value, clock, gates, and mode states from the shared normalized mixer input using channel-local operations, the channels remain independent. If a truncated model keeps $1, \cdots, K$ channels and discards higher-indexed channels, the output becomes

$$\hat{y}(t) = K^{-1/2} \textstyle\sum_{k=1}^{K} W^{\mathrm{out}}_k z_k(t), \qquad W^{\mathrm{out}}_k \in \mathbb{R}^{d \times P}.$$

Channel truncation changes both the terms included in the sum and its $K^{-1/2}$ normalization, so the truncated mixer does not generally produce the same output as the full mixer. Next, we introduce methods to train these reduced-capacity models to maintain model performance after truncation.

4.2 Two-Rate Elastic Training Across the Hybrid

Capacity tiers and model truncation. We define a capacity tier T by its retained spectral-channel count K_T and feed-forward width $n_{\mathrm{ff},T}$. These two quantities control different parts of deployment cost: the spectral-channel count determines recurrent-state size and mixer computation, whereas the feed-forward layer contains most of a spectral block's parameters (Table 4). We reduce these two capacity axes at different rates, retaining a larger fraction of spectral channels while reducing feed-forward width more aggressively. We encode the two rates with a budget variable $\xi_T \in (0, 1]$:

$$K_T = \max\{1, \mathrm{round}(\bar{K} \xi_T^{1/2})\}, \qquad n_{\mathrm{ff},T} = 64 \max\left\{1, \left\lfloor \frac{d_{\mathrm{ff}} \xi_T^{0.8}}{64} \right\rfloor \right\}. \tag{7}$$

Here, $\bar{K}$ is the full spectral-channel count, K_T is the number of channels retained by tier T, d_{ff} is the full feed-forward width and $n_{\mathrm{ff},T}$ is its retained width. At tier T, channel truncation retains spectral channels $1, \ldots, K_T$ and discards channels $k > K_T$. Similarly, each feed-forward layer retains the first $n_{\mathrm{ff},T}$ units under the importance ordering. The budget variable ξ_T determines the retained tensor shapes rather than the exact exported parameter fraction. Because $\xi_T^{1/2} > \xi_T^{0.8}$ for $0 < \xi_T < 1$, reduced-capacity tiers retain a larger fraction of spectral channels than feed-forward units. Since the feed-forward layer also contains most of a spectral block's parameters, most of the parameter reduction occurs in the feed-forward layers, while recurrent-state size and mixer computation decrease more gradually. Attention, embeddings, and residual width do not vary with tier. The exponents are empirical design choices evaluated in Appendix M.1. Table 6 gives the resulting model sizes.

Capacity-mixed elastic training. The two-rate map defines the spectral-channel count and feed-forward width of each tier, but it does not ensure that reduced-capacity models maintain model

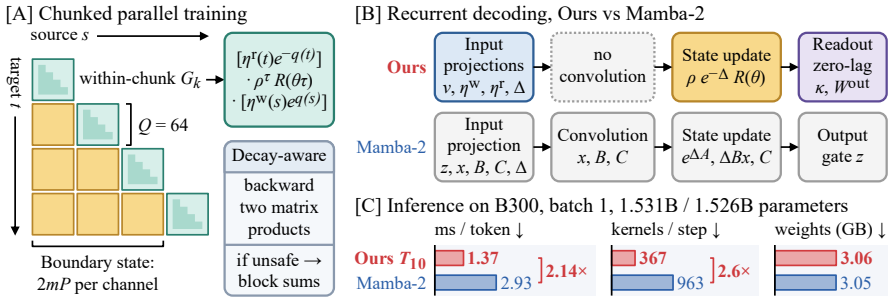

Figure 2: Efficient execution of the selective mixer. (A) A chunked parallel scan evaluates within-chunk contributions in parallel, while mode states connect consecutive chunks. (B) Mixer-level recurrent decoding is compared with Mamba-2. (C) Inference efficiency of ESSH versus parameter-matched Mamba-2 on B300. Full efficiency results: Appendices P and R.

performance. To address this, we introduce capacity-mixed training, in which each optimizer step combines full-capacity micro-batches with micro-batches assigned to sampled reduced capacities. For a reduced-capacity micro-batch, the full-capacity tier T_{10} is evaluated on the same input and supplies a stopped-gradient distillation target (Hinton et al., 2015):

$$\mathcal{L}_T = \mathrm{CE}(p_T, \mathrm{id}_{t+1}) + 0.5\,\mathrm{KL}\big(\mathrm{sg}[p_{T_{10}}] \,\|\, p_T\big), \tag{8}$$

where p_T and $p_{T_{10}}$ are the next-token distributions produced by tier T and the full-capacity tier, respectively, and $\mathrm{sg}[\cdot]$ denotes stop-gradient. The cross-entropy (CE) term is the negative log-likelihood of the observed next token, while the KL divergence encourages the reduced-capacity distribution to match the full-capacity distribution. If $T = T_{10}$, the KL divergence is omitted.

Because each reduced-capacity tier retains channels $1, \ldots, K_T$, lower-index channels are included in more sampled reduced-capacity micro-batches. Consequently, they are optimized under a broader range of reduced-capacity configurations, which follows the nested-dropout principle (Rippel et al., 2014; Song et al., 2026). For retained feed-forward weights, gradients from reduced-capacity micro-batches are additionally scaled by $\alpha_{\mathrm{ff}} = 0.5$, which reduces their contribution to the shared parameter update without changing the forward computation. Together with the two-rate capacity map, our elastic-training recipe uses budget dropout (the capacity sampling above) over capacity tiers, full-tier distillation for reduced-capacity models, feed-forward importance ordering, and reduced-capacity feed-forward gradient scaling. Appendices D.3 and D.4 give the details for these. After pretraining and a brief shared refinement stage (Appendix D.8), each tier is exported as a standalone model by slicing its spectral-channel and feed-forward tensors.

4.3 Efficient Parallel Training and Recurrent Decoding

The structured transition in (6) supports both parallel training and recurrent decoding (Figure 2) using the same selective recurrence.

Chunked parallel training. During training, we divide each input sequence into contiguous chunks of at most $Q = 64$ positions. Within a chunk, the closed-form transition in (6) allows the contributions between all source and target positions to be evaluated in parallel. Specifically, for channel k, stacking the chunk's channel values into V_k and its channel readouts into Z_k^{core} gives $Z_k^{\mathrm{core}} = G_k V_k + B_k$, where G_k contains the causal within-chunk contributions and B_k contains the contributions carried by the entering mode states. For positions $b \le s \le t \le e$ of a chunk $[b, e]$, one has $G_k[t, s] = \sum_{i=1}^{m} \eta^{r}_{k,i}(t)\eta^{w}_{k,i}(s)\kappa^{\top}_{k,i}\Pi_{k,i}(t, s)e_1$, and $G_k[t, s] = 0$ for $s > t$. Thus, the within-chunk computation is performed by matrix multiplication, while only $2mP$ recurrent-state entries per channel are carried between consecutive chunks (Dao & Gu, 2024). The exact construction of G_k, the boundary-state update, and the backward pass are given in Appendix O.1–O.2.

Recurrent decoding. During recurrent decoding, the same mode states are updated one token at a time. We use a fused recurrent path that combines the retained projections, state updates, and readouts, without evaluating discarded channels. At tier T, each spectral layer stores $2K_T mP$ recurrent-state entries, while sliding-window attention contributes a fixed-size key–value cache.

Table 1: Quality and inference cost: (a) 1.5B full models; (b) 1.5B ESSH tiers; (c) 370M models. Pretraining uses 100B FineWeb-Edu tokens in (a,b) and 7.4B in (c), with ESSH tiers sharing training. †: published values from Table 1 of Lahoti et al. (2026). PPL uses each source's validation shard. Accuracies: percentages in (a,b), fractions in (c); full-key retrieval uses 2K contexts. B300 inference in (b) uses batch one and 2K context. Protocols: Appendix E. Results: Appendices F–H, N.1, and R.

(a) Model	Params (B)	PPL ↓	LAMB	HellaS	PIQA	ARC-e	ARC-c	Wino	OBQA	Mean
ESSH	1.53	**10.11**	46.8	60.6	72.9	73.7	40.7	58.3	32.2	55.0
Transformer++†	∼ 1.5	10.51	50.3	60.6	73.8	74.0	40.4	58.7	29.6	55.4
Gated DeltaNet†	∼ 1.5	10.45	49.2	61.3	74.3	75.3	41.2	58.0	31.6	55.8
Mamba-2†	∼ 1.5	10.47	47.8	61.4	73.6	75.3	41.8	57.5	**32.6**	55.7
Mamba-3 SISO†	1.49	10.35	49.4	61.9	73.6	75.9	42.7	59.4	32.0	56.4
Mamba-3 MIMO†	1.50	10.24	**51.7**	**62.3**	**75.3**	**76.5**	**44.5**	**60.6**	**32.6**	**57.6**

(b) Metric	T_1	T_4	T_7	T_{10}
Params (B)	0.474	0.828	1.168	1.531
PPL ↓	16.49	12.26	10.97	**10.11**
7-task ↑	43.8	50.7	53.0	**55.0**
Full key ↑	37.0	**37.9**	33.9	33.0
Decode (ms) ↓	**1.13**	1.20	1.34	1.37
Prefill (ktok/s) ↑	75.0	**81.4**	74.0	71.0
State+KV (MB) ↓	**51.15**	52.17	52.89	53.61

(c) Model	Params (M)	7-task ↑	Full key ↑
ESSH T_1	92.1	0.376	0.241
ESSH T_4	185.3	0.405	0.370
ESSH T_7	274.3	0.420	0.366
ESSH T_{10}	370.9	0.428	**0.389**
Mamba-3 + SWA	377.1	**0.445**	0.230
Gated DeltaNet + SWA	378.1	0.433	0.100
Transformer++	365.6	0.414	0.350
MatFormer	365.6	0.411	0.266
MatMamba	369.9	0.404	0.004

5 EXPERIMENTS

We evaluate ESSH on model quality, elastic quality–size trade-offs, and inference efficiency. We use a 1.5B-scale study to assess large-scale quality and deployment performance. The model is pretrained on 100B FineWeb-Edu tokens with the Llama-3.1 tokenizer (Penedo et al., 2024; Grattafiori et al., 2024). We also use a 370M-scale study for controlled comparisons with elastic baselines, independently trained models, and ablations. It uses 7.4B FineWeb-Edu tokens with the GPT-2 tokenizer. One-change ablations use 3.7B tokens. Both models use a training context of 2048 tokens. Appendix E gives configurations and protocols. Training costs are reported in Appendices N.7 and N.8. Evaluation includes FineWeb-Edu perplexity, a seven-task zero-shot language-understanding suite, NIAH retrieval, and a 2K–16K context-length sweep. We measure the training cost under four deployment sizes and benchmark decode latency and throughput, prefill throughput, and persistent state+KV memory.

5.1 MAIN-MODEL EVALUATION

As shown in Table 1(a), the 1.53B ESSH model achieves a seven-task mean of 55.03%, within 0.37 percentage points of the published Transformer++ reference and 0.67 points of Mamba-2. This competitiveness is also reflected across individual tasks: ESSH matches Transformer++ on HellaSwag at 60.6% and scores higher on ARC-Challenge (40.7% versus 40.4%) and OpenBookQA (32.2% versus 29.6%). ESSH also reaches a validation perplexity (PPL) of 10.11 on next-token prediction, remaining competitive at full capacity while jointly training reduced-capacity models.

It is worth noting that elasticity introduces a trade-off between full-capacity and compact-model quality. After warm-up, 77.5% of micro-batches use full capacity in expectation, while the remaining updates are devoted to reduced-capacity tiers through the shared parameters. It is therefore reasonable that the full-capacity model does not match the best independently optimized baselines, since part of the training budget is used to improve compact exports. To further validate this trade-off, we conduct a controlled 370M-scale study: allocating more training to the full-capacity model improves its PPL but degrades the smallest tier (Appendices M.2–M.3).

On the other hand, the benefit of the aforementioned trade-off is that the same 1.5B-scale training run produces a family of standalone models at smaller sizes. The 0.828B ESSH-T_4 export retains

a seven-task mean of 50.72% with 45.9% fewer parameters, while ESSH-T_7 reaches 53.01% at 1.168B parameters (Table 1(b)). Across the ten evaluated tiers, PPL increases smoothly as capacity decreases. This provides a controllable quality–size trade-off from a single training run.

5.2 The Elastic Frontier: Quality–Capacity Trade-offs

Comparison with elastic baselines (Figure 3(a)). At the 370M scale, we compare ESSH with elastic baselines MatFormer and MatMamba to evaluate quality as deployment capacity decreases. From Figure 3(a), ESSH has a better PPL–size trade-off at intermediate sizes: the 214.5M ESSH-T_5 matches full MatMamba's PPL of 16.98 with 42.0% fewer parameters. At nearly matched model sizes, the 274.3M ESSH-T_7 achieves a lower PPL of 15.91 than 16.98 for the 270.9M MatFormer.

Shared training versus independent models (Figure 3(b)–(c)) Next, we compare ESSH exports with independently trained Transformer++ models at corresponding deployment sizes in Figure 3(b). ESSH has the higher seven-task mean at three of the four sizes (by up to $+1.46$ points) and trails only at T_1 (-0.85). At ESSH-T_7, for example, ESSH obtains both a higher seven-task mean (42.00% versus 40.79%) and lower perplexity (15.91 versus 16.69) at nearly the same parameter count.

Furthermore, as shown in Figure 3(c), producing these four ESSH models requires a single shared training run of an estimated 53.9 A100-hours, 46.9% less than the 101.4 A100-hours of four independently trained Transformer++ models. This advantage grows as more deployment sizes are required because additional ESSH tiers are exported from the same trained family without retraining.

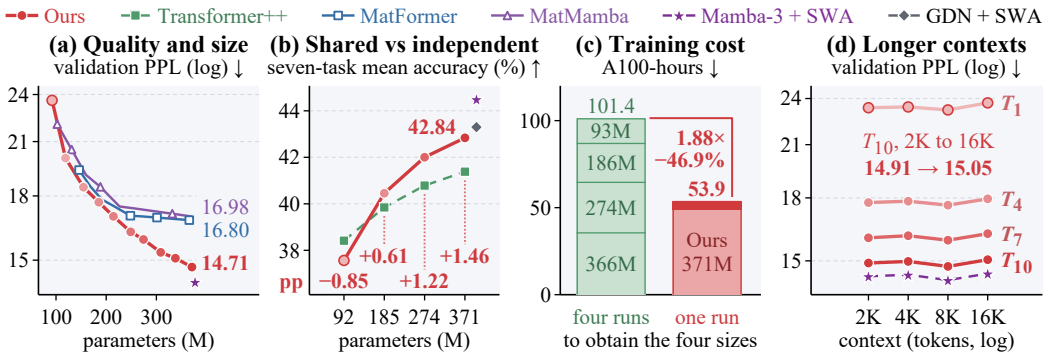

Figure 3: Quality and training cost at the 370M scale. Connected points belong to one model family, isolated markers are single full-size models, and T_k denotes an ESSH tier. (a) PPL versus model size; (b) seven-task accuracy against independently trained Transformer++; (c) estimated A100-hours to obtain four sizes from four Transformer++ runs (green) or one shared 371M ESSH run (red), each labelled by size; (d) PPL at longer contexts. Protocols and full results: Appendices N.1, N.8, and H.3.

Retrieval and longer contexts (Figure 3(d) and Table 1(c)). We evaluate whether reduced-capacity exports retain retrieval capability and whether their degradation increases beyond the 2K training context. As shown in Table 1(c), on the 2K NIAH retrieval test, full ESSH achieves a full-key accuracy of 0.389, compared with 0.350 for Transformer++ and 0.230 for Mamba-3+SWA. The half-size T_4 retains 0.370. This retention is consistent with the fixed attention layout: the supplied key remains directly accessible while spectral-channel count and feed-forward width decrease. Aggressive parameter reduction to T_1 lowers accuracy to 0.241.

Figure 3(d) further shows that capacity reduction does not substantially amplify PPL degradation beyond the training context. From 2K to 16K tokens, the T_4–T_{10} PPL gap remains nearly constant. Each tier carries its recurrent state across tokens while attention stays within the same local window.

Mechanism ablations. Matched 3.7B-token ablations isolate the two main components of ESSH (Appendices L–M). First, making the clock input-independent or neutralizing the write/read gates raises PPL at every tier and lowers T_4 retrieval accuracy, showing that selective state transition and selective writing/readout contribute beyond the fixed spectral recurrence. Second, elastic training

is necessary for aggressive truncation: at T_1, a model trained only at full capacity, then truncated and refined like ESSH, reaches PPL 45.66, compared with 25.32 for ESSH. Removing full-tier distillation increases T_1 PPL from 25.32 to 26.42. These results support the importance of both selective dynamics and capacity-mixed training.

5.3 Inference Efficiency

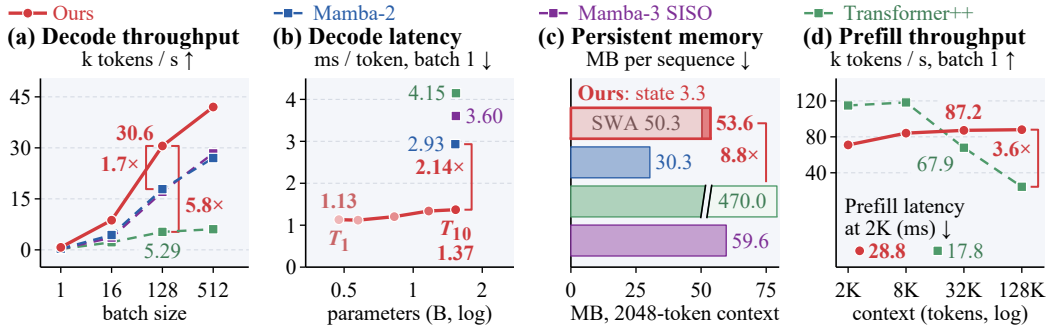

Figure 4: B300 deployment at matched full-model parameter counts (1.526B–1.536B). (a) Decode throughput; (b) batch-one latency; (c) state+KV per sequence; (d) batch-one prefill. Decode uses a 2K context, and compact ESSH tiers use their sizes. Matched full-model values are three-session medians; compact prefill uses single-session measurements; full measurements are in Appendix R.

Recurrent decoding. As discussed in Sec. 4.3, we validate whether the structured recurrence provides an execution benefit in addition to supporting channel-wise truncation. At matched full-model parameter counts, ESSH has the lowest decode latency at all four measured B300 batch sizes (Figure 4). At batch-one, ESSH takes 1.37 ms/token, which is a 2.14–2.80× speedup over the tested Mamba implementations and 3.03× over Transformer++. ESSH also has the lowest decode latency in the matched H200 measurements. We note that this decoding advantage is not caused by elastic model reduction but by the structured computation and fused implementation. ESSH and Mamba-2 use 3.06 and 3.05 GB of memory for model weights, respectively, but execute 367 and 963 compute kernels per decoding step (Figure 2(C)). ESSH projects each channel value once and reuses it across its modes, enabling the projections, recurrent updates, and readouts to be fused. A same-architecture control on A100 further confirms this implementation benefit: replacing the unfused execution path with the fused path reduces batch-one latency from 7.40 to 3.02 ms.

Memory and long-context prefill. ESSH also requires substantially less persistent sequence memory than full attention. At a 2K context, the 1.53B model uses 53.61 MB of recurrent state plus KV cache per sequence, compared with 469.99 MB for Transformer++. Most of the ESSH memory consumption comes from the fixed sliding-window attention cache, while reducing the spectral-channel count further decreases the recurrent-state memory of compact tiers. For prefill, Transformer++ is faster at 2K tokens, but its cost increases more rapidly with context length. At 128K tokens, ESSH processes 88.0 ktok/s compared with 24.4 ktok/s for Transformer++, because the spectral scan and sliding-window attention avoid full-prefix attention.

6 Conclusion

ESSH extends LTI spectral filters to input-dependent recurrences while preserving channel-wise truncation. Capacity-mixed training with budget dropout and full-tier distillation maintains model quality across reduced-capacity tiers, while the two-rate capacity map jointly controls spectral-channel count and FFN width. Experiments show that compact exports retain substantial task and retrieval performance, and that the structured recurrence supports chunked parallel training and fused recurrent decoding with lower measured latency at matched model sizes. As a limitation, the current framework varies capacity through spectral-channel count and FFN width while keeping the attention layout, depth, and residual width fixed. Extending elasticity to these shared components and developing capacity-selection strategies under model-quality and device constraints are natural directions for future work.

AI USE STATEMENT

Generative AI tools were used for language editing and manuscript polishing. All AI-assisted edits were reviewed by the authors, who take responsibility for the final content.

REPRODUCIBILITY STATEMENT

Appendices C and D specify the model parameterization and training procedure, and Appendix E describes the data, baselines, and evaluation protocols. Appendices B and O provide the derivations and systems algorithms. Appendix Q records the hyperparameters, export procedure, evaluation environment, and compute accounting. The supplementary material includes code for model implementation, training, ablations, evaluation, standalone export, and benchmarking.

REFERENCES

Naman Agarwal, Daniel Suo, Xinyi Chen, and Elad Hazan. Spectral state space models. *arXiv preprint arXiv:2312.06837*, 2023.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Y Zou, Atri Rudra, and Christopher Re. Zoology: Measuring and improving recall in efficient language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 15664–15730, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/448fc91f669c15d10364ee01d512cc10-Paper-Conference.pdf.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. Simple linear attention language models balance the recall-throughput tradeoff. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 1763–1840. PMLR, 21–27 Jul 2024b. URL https://proceedings.mlr.press/v235/arora24a.html.

Simran Arora, Aman Timalsina, Aaryan Singhal, Benjamin Spector, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Ré. Just read twice: closing the recall gap for recurrent language models, 2024c. URL https://arxiv.org/abs/2407.05483.

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes, 2025. URL https://arxiv.org/abs/2304.09433.

Zhangir Azerbayev, Edward Ayers, and Bartosz Piotrowski. Proof-Pile. Hugging Face dataset repository, 2022. URL https://huggingface.co/datasets/hoskinson-center/proof-pile. Accessed 2026-09-26.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7432–7439. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HylxE1HKwS.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5298–5311. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/cai24e.html.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019.

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.

CodeParrot. Github code dataset. Hugging Face dataset card, 2022. URL https://huggingface.co/datasets/codeparrot/github-code. Accessed 2026-09-08.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. Matformer: Nested transformer for elastic inference. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 140535–140564. Curran Associates, Inc., 2024. doi: 10.52202/079017-4461. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/fe066022bab2a6c6a3c57032a1623c70-Paper-Conference.pdf.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, 2019.

Felix R Gantmacher and Mark G Krein. *Oscillation matrices and kernels and small vibrations of mechanical systems: revised edition*. American Mathematical Society Rhode Island, 2002.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2020. URL https://arxiv.org/abs/2101.00027.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7b ssm hybrid model, 2024. URL https://arxiv.org/abs/2405.16712.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab

AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi (Jack) Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias

Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu (Sid) Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Katarína Grešová, Vlastimil Martinek, David Čechák, Petr Šimeček, and Panagiotis Alexiou. Genomic benchmarks: a collection of datasets for genomic sequence classification. *BMC Genomic Data*, 24(1):25, 2023.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 35971–35983. Curran Associates, Inc., 2022. doi: 10.52202/068431-2607. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/e9a32fade47b906de908431991440f7c-Paper-Conference.pdf.

Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. news-please: A generic news crawler and extractor. In *Proceedings of the 15th International Symposium of Information Science*, pp. 218–223, March 2017. doi: 10.5281/zenodo.4120316.

Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 775–784, 2011.

Elad Hazan, Karan Singh, and Cyril Zhang. Learning linear dynamical systems via spectral filtering. *Advances in Neural Information Processing Systems*, 30, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Yingbo Hua and Tapan K. Sarkar. Matrix pencil method for estimating parameters of exponentially damped/undamped sinusoids in noise. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(5):814–824, 1990. doi: 10.1109/29.56027.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio

Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 1601–1611, 2017. doi: 10.18653/v1/P17-1147.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2205.13147.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

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 C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 86173–86202, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/8abd2043b71a074278d5f687947bff9c-Paper-Conference.pdf.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, 2017. doi: 10.18653/v1/D17-1082.

Barak Lenz, Opher Lieber, Alan Arazi, Amir Bergman, Avshalom Manevich, Barak Peleg, Ben Aviram, Chen Almagor, Clara Fridman, Dan Padnos, Daniel Gissin, Daniel Jannai, Dor Muhlgay, Dor Zimberg, Edden M. Gerber, Elad Dolev, Eran Krakovsky, Erez Safahi, Erez Schwartz, Gal Cohen, Gal Shachaf, Haim Rozenblum, Hofit Bata, Ido Blass, Inbal Magar, Itay Dalmedigos, Jhonathan Osin, Julie Fadlon, Maria Rozman, Matan Danos, Michael Gokhman, Mor Zusman, Naama Gidron, Nir Ratner, Noam Gat, Noam Rozen, Oded Fried, Ohad Leshno, Omer Antverg, Omri Abend, Or Dagan, Orit Cohavi, Raz Alon, Ro'i Belson, Roi Cohen, Rom Gilad, Roman Glozman, Shahar Lev, Shai Shalev-Shwartz, Shaked Haim Meirom, Tal Delbari, Tal Ness, Tomer Asida, Tom Ben Gal, Tom Braude, Uriya Pumerantz, Josh Cohen, Yonatan Belinkov, Yuval Globerson, Yuval Peleg Levy, and Yoav Shoham. Jamba: Hybrid transformer-mamba language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=JFPaD7lpBD.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. LogiQA: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 3622–3628, 2020. doi: 10.24963/ijcai.2020/501.

Y. Isabel Liu, Windsor Nguyen, Yagiz Devre, Evan Dogariu, Anirudha Majumdar, and Elad Hazan. Flash stu: Fast spectral transform units. In *Proceedings of the 2025 IEEE 64th Conference on Decision and Control (CDC)*, pp. 165–171, 2025. arXiv:2409.10489.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe. Introduces WikiText-2 and WikiText-103; arXiv:1609.07843.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018. doi: 10.18653/v1/D18-1260.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 41076–41102. Curran Associates, Inc., 2024. doi: 10.52202/079017-1299. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/4822991365c962105b1b95b1107d30e5-Paper-Conference.pdf.

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 *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 1525–1534, 2016. doi: 10.18653/v1/P16-1144.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

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 A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylKikSYDH. Introduces the PG-19 benchmark; arXiv:1911.05507.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, 2016.

Oren Rippel, Michael A. Gelbart, and Ryan P. Adams. Learning ordered representations with nested dropout. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, 2011. URL https://cdn.aaai.org/ocs/2418/2418-10878-1-PB.pdf.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 4463–4473, 2019. doi: 10.18653/v1/D19-1454.

Yair Schiff, Chia Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range DNA sequence modeling. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 43632–43648, 2024. URL https://proceedings.mlr.press/v235/schiff24a.html.

Valerie A Schneider, Tina Graves-Lindsay, Kerstin Howe, Nathan Bouk, Hsiu-Chuan Chen, Paul A Kitts, Terence D Murphy, Kim D Pruitt, Françoise Thibaud-Nissen, Derek Albracht, et al. Evaluation of grch38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome research*, 27(5):849–864, 2017.

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

Abhinav Shukla, Sai Vemprala, Aditya Kusupati, and Ashish Kapoor. Matmamba: A matryoshka state space model. *arXiv preprint arXiv:2410.06718*, 2024.

Dachuan Song, Junyu Yin, Zechen Hu, and Xuan Wang. Elastic spectral state space models for train-once budgeted inference, 2026. URL https://arxiv.org/abs/2601.22488.

State Space Models. Mamba-3 MIMO 1.50 B: Official model card, 2026a. URL https://huggingface.co/state-spaces/mamba3-mimo-1.5b.

State Space Models. Mamba-3 SISO 1.49 B: Official model card, 2026b. URL https://huggingface.co/state-spaces/mamba3-siso-1.5b.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.127063. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864.

Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, ben fattori, Charles Lovering, farzanehnakhaee70, Jason Phang, Anish Thite, Fazz, Aflah, Niklas, Thomas Wang, sdtblck, nopperl, gakada, tttyuntian, researcher2, Julen Etxaniz, Chris, Hanwool Albert Lee, Leonid Sinev, Zdeněk Kasner, Kiersten Stokes, Khalid, KonradSzafer, Jeffrey Hsu, and Anjor Kanekar. EleutherAI/lm-evaluation-harness: v0.4.9.1, August 2025. URL https://doi.org/10.5281/zenodo.16737642.

Ali Taghibakhshi, Sharath Turuvekere Sreenivas, Saurav Muralidharan, Ruisi Cai, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Yoshi Suhara, Oluwatobi Olabiyi, Daniel Korzekwa, Mostofa Patwary, Mohammad Shoeybi, Jan Kautz, Bryan Catanzaro, Ashwath Aithal, Nima Tajbakhsh, and Pavlo Molchanov. Nemotron elastic: Towards efficient many-in-one reasoning llms, 2025a. URL https://arxiv.org/abs/2511.16664.

Ali Taghibakhshi, Sharath Turuvekere Sreenivas, Saurav Muralidharan, Marcin Chochowski, Yashaswi Karnati, Raviraj Joshi, Ameya Mahabaleshwarkar, ZIJIA CHEN, Yoshi Suhara, Oluwatobi Olabiyi, Daniel Korzekwa, Mostofa Patwary, Mohammad Shoeybi, Jan Kautz, Bryan Catanzaro, Ashwath Aithal, Nima Tajbakhsh, and Pavlo Molchanov. Efficient hybrid language model compression through group-aware ssm pruning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 122956–122976. Curran Associates, Inc., 2025b. doi: 10.52202/085713-4100. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/b2438954d8c3cad59b3ef433b5c15072-Paper-Conference.pdf.

Ali Taghibakhshi, Ruisi Cai, Saurav Muralidharan, Sharath Turuvekere Sreenivas, Aditya Vavre, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Sheldon Liang, Marcin Chochowski, Zijia Chen, Akhiad Bercovich, Ran Zilberstein, Ran El-Yaniv, Yonatan Geifman, Daniel Korzekwa, Yoshi

Suhara, Oluwatobi Olabiyi, Ashwath Aithal, Nima Tajbakhsh, and Pavlo Molchanov. Star elastic: Many-in-one reasoning llms with efficient budget control, 2026. URL https://arxiv.org/abs/2605.07182.

Philippe Tillet, H. T. Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, pp. 10–19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367196. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

Aaron Voelker, Ivana Kajić, and Chris Eliasmith. Legendre memory units: Continuous-time representation in recurrent neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/952285b9b7e7a1be5aa7849f32ffff05-Paper.pdf.

Yaoxiang Wang, Qingguo Hu, Yucheng Ding, Ruizhe Wang, Yeyun Gong, Jian Jiao, Yelong Shen, Peng Cheng, and Jinsong Su. Training matryoshka mixture-of-experts for elastic inference-time expert utilization, 2025. URL https://arxiv.org/abs/2509.26520.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, 2017. doi: 10.18653/v1/W17-4413.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 29687–29707, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/4904fad153f6434a7bcf04465d4be2cc-Paper-Conference.pdf.

Jiahui Yu and Thomas Huang. Universally slimmable networks and improved training techniques. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1812.08928.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019. doi: 10.18653/v1/P19-1472.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

APPENDIX

A Notation 19
B Derivations 21
C Mixer Parameterization and Cost 27
D Elastic Training 30
E Experimental Protocols 34
F Complete Main Results 38
G Downstream Language Understanding 40
H Retrieval and Length 41
I Synthetic Recall Suite 44
J Cross-Domain Language Modeling 46
K DNA Sequence Modeling 47
L Selective-Dynamics Ablations 48
M Elasticity Ablations 53
N Compression and Train-Once Baselines 57
O Systems: Algorithms and Kernels 62
P Systems Evaluation 68
Q Reproducibility 72
R Cross-Device Efficiency 73

A NOTATION

Tables 2 and 3 summarize notation for the generic SSM, the selective spectral mixer, and the elastic capacity tiers. Channel values and channel outputs are column vectors in $\mathbb{R}^P$, while each mode state lies in $\mathbb{R}^{2\times P}$; the collection of retained mode states forms the recurrent state. Feed-forward network is abbreviated as FFN. Throughout, *state update* refers to the recurrent dynamics, *temporal retention* to the persistence of earlier contributions through those dynamics, and *memory* to hardware storage.

The index k denotes a spectral channel and i a mode within that channel. A capacity tier T retains channels $1, \ldots, K_T$ and feed-forward width $n_{\mathrm{ff},T}$. The two-rate map determines both quantities from the budget variable ξ_T, while the reported parameter fraction r is measured from the resulting standalone export. Exporting the corresponding tensor slices produces a standalone model. In a metric name, Δ denotes a difference; clock increments carry channel and mode indices. Indices introduced within a proof or algorithm are local to that construction.

Table 2: Notation for the generic SSM and Hankel spectral construction.

Symbol	Meaning
$L, t, s, \tau = t - s$	sequence length, target and source positions, and lag
$u(t)$, $h(t)$, $y(t)$	input, latent state, and output of a generic state-space model
A, B, C, D	state transition, input, readout, and direct input–output matrices
N, j	LTI state dimension and eigenvalue index
$d_{\mathrm{in}}, d_{\mathrm{out}}$	input and output dimensions of the generic SSM
$\mathcal{P}_K$, Γ_j	rank-K Hankel projection and spectral residue matrix
U, Λ, λ_j	eigenbasis, diagonal eigenvalue matrix, and one eigenvalue of A
$G(\tau)$, $G(t, s)$	LTI kernel and its time-varying counterpart
$\nu(\lambda)$, $\mu(\lambda)$, Z	geometric response, weighted response, and Hankel matrix
(σ_k, ϕ_k), Φ_k, M_k	Hankel eigenpair, corresponding causal filter, and learned channel projection
K, k	retained spectral-channel count and spectral-channel index

Table 3: Notation for the selective spectral mixer and elastic training.

Symbol	Meaning
$u(t)$	mixer input after RMSNorm
d, d_{ff}	residual width and full FFN hidden width
$\bar{K}$, K_T	full spectral-channel count and channel count retained by tier T, with $\bar{K} = 32$ in the default recipe
m, i, P	modes per channel, mode index, and channel value width; default language models use $m = 8$ and $P = d/\bar{K}$
$H_{k,i}(t)$	mode state in $\mathbb{R}^{2\times P}$
$v_k^0(t)$, $v_k(t)$	projected channel value before and after the channel-local normalization blend
$z_k^{\mathrm{core}}(t)$, $z_k(t)$	core channel readout and channel output after the zero-lag correction, both in $\mathbb{R}^P$
W_k^{in}, W_k^{out}	channel input and output projections, of shapes $P \times d$ and $d \times P$
e_1, $\kappa_{k,i}$	$(1,0)^\top$ and $(\kappa_{k,i}^{\mathrm{c}}, -\kappa_{k,i}^{\mathrm{s}})^\top$
$A_{k,i}(t)$, $R(\theta)$	input-dependent mode transition and planar rotation
$\rho_{k,i}$, $\gamma_{k,i} = -\log \rho_{k,i}$, $1/\gamma_{k,i}$	intrinsic decay factor, decay rate, and intrinsic e-folding horizon
$\theta_{k,i}$, $\kappa_{k,i}^{\mathrm{c}}$, $\kappa_{k,i}^{\mathrm{s}}$	rotation frequency and modal readout coefficients
$\eta_{k,i}^{\mathrm{w}}(t)$, $\eta_{k,i}^{\mathrm{r}}(t)$	write and read gates
$\Delta_{k,i}(t)$, $\mathrm{e}^{-\Delta_{k,i}(t)}$, $q_{k,i}(t)$	clock increment, exponential factor, and accumulated clock, with $q_{k,i}(0) = 0$
$\Pi_{k,i}(t,s)$	structured transition product from position s to t
G_k, V_k, B_k, Z_k^{core}	within-chunk causal interaction matrix, stacked values, entering-state contribution, and stacked core readouts
$\xi_T \in (0,1]$	input budget variable of the two-rate capacity map for configuration T
r	parameter fraction of the standalone export relative to the full model
T_1–T_{10}, T	the ten evaluated deployment tiers and a generic capacity configuration; T_{10} is full capacity
$n_{\mathrm{ff},T}$	FFN hidden width retained by tier T
$f = 0.75$, $\alpha_{\mathrm{ff}} = 0.5$	reserved full-capacity micro-batch fraction and reduced-capacity FFN gradient scale
α_{mix}	reduced-capacity mixer-gradient scale used in the optimization ablation
p_T, $p_{T_{10}}$	next-token distributions of tier T and the full-capacity tier
Θ, Ω	model parameters and optimizer state
β_k, $a_{k,i}$	channel-local normalization-blend coefficient and mode zero-lag correction coefficient
$\widehat{\phi}_k^{(m)}(\tau)$, $\psi_{k,i}(\tau)$	fitted channel response and individual damped mode response
$Q = 64$, b, e	maximum chunk length and first and last positions of a chunk

B DERIVATIONS

This section develops the mathematical results underlying the spectral construction, the selective recurrence, and channel truncation. These results characterize the fixed spectral representation and the realized selective dynamics; prediction quality after training and truncation is evaluated empirically.

We use $z_k^{\mathrm{core}}(t)$ for the sum of the mode readouts and $z_k(t)$ for the channel output after the zero-lag correction in Equation 24. Both are column vectors in $\mathbb{R}^P$ before the output projection. With $H_{k,i}(0) = 0$, $e_1 = (1, 0)^\top$, and $\kappa_{k,i} = (\kappa^{\mathrm{c}}_{k,i}, -\kappa^{\mathrm{s}}_{k,i})^\top$, the core recurrence is

$$\begin{aligned} H_{k,i}(t) &= \rho_{k,i} \mathrm{e}^{-\Delta_{k,i}(t)} R(\theta_{k,i}) H_{k,i}(t-1) + \eta^{\mathrm{w}}_{k,i}(t) e_1 v_k(t)^\top, \\ z_k^{\mathrm{core}}(t) &= \sum_i \eta^{\mathrm{r}}_{k,i}(t) H_{k,i}(t)^\top \kappa_{k,i}. \end{aligned} \tag{9}$$

Here $0 < \rho_{k,i} < 1$, $\Delta_{k,i}(t) > 0$, and the write/read gates lie in $(0, 2)$. The intrinsic decay rate is $\gamma_{k,i} = -\log \rho_{k,i} > 0$. Unless stated otherwise, the derivations below use the core output; Appendix C.3 shows how the zero-lag correction modifies the lag-zero term.

B.1 MODAL FORM OF REAL RECURRENCES

A complex-conjugate pole pair in a real SSM can be represented by a two-dimensional damped rotation; related SSM representations are discussed by Lahoti et al. (2026). It is sufficient here to consider the scalar temporal response, because each fitted mode applies the same temporal dynamics independently to the P coordinates of a channel value. Here j denotes the imaginary unit, with $\mathrm{j}^2 = -1$.

Let $A \in \mathbb{R}^{N \times N}$ be diagonalizable over $\mathbb{C}$, with spectral radius less than one, and let $B \in \mathbb{R}^N$ and $C \in \mathbb{R}^{1 \times N}$. Write $A = U \Lambda U^{-1}$ and define

$$r_j = (CU)_j (U^{-1} B)_j.$$

For integer $\tau \geq 0$,

$$CA^\tau B = \sum_j r_j \lambda_j^\tau.$$

Because A, B, and C are real, every nonreal eigenvalue and its residue occur with their complex conjugates. For a conjugate pair $\rho_i \mathrm{e}^{\pm \mathrm{j}\theta_i}$, write

$$r_i = \frac{\kappa^{\mathrm{c}}_i + \mathrm{j}\kappa^{\mathrm{s}}_i}{2}.$$

The pair contributes

$$\begin{aligned} &r_i (\rho_i \mathrm{e}^{\mathrm{j}\theta_i})^\tau + \overline{r_i} (\rho_i \mathrm{e}^{-\mathrm{j}\theta_i})^\tau \\ &\qquad = \rho_i^\tau \big(\kappa^{\mathrm{c}}_i \cos(\theta_i \tau) - \kappa^{\mathrm{s}}_i \sin(\theta_i \tau) \big). \end{aligned}$$

Thus the impulse response can be written as

$$CA^\tau B = \sum_{j \in \mathcal{R}} r_j \lambda_j^\tau + \sum_{i \in \mathcal{C}} \rho_i^\tau \big(\kappa^{\mathrm{c}}_i \cos(\theta_i \tau) - \kappa^{\mathrm{s}}_i \sin(\theta_i \tau) \big),$$

where $\mathcal{R}$ indexes real poles and $\mathcal{C}$ indexes nonreal conjugate pairs.

The same conjugate-pair term has a real two-dimensional recurrence. An impulse injected along e_1 evolves as

$$(\rho_i R(\theta_i))^\tau e_1 = \rho_i^\tau \big(\cos(\theta_i \tau), \sin(\theta_i \tau) \big)^\top.$$

Reading this state with $(\kappa^{\mathrm{c}}_i, -\kappa^{\mathrm{s}}_i)$ gives the expression above. A nonzero real pole is included by taking $\rho = |\lambda|$ with $\theta = 0$ for $\lambda > 0$ and $\theta = \pi$ for $\lambda < 0$; a zero pole contributes only at lag zero and can be handled separately.

Exact representation requires enough modes to cover the active poles and excludes nontrivial Jordan blocks, whose responses contain polynomial factors in τ. The fitted bank uses m modes per channel under these representation conditions.

B.2 Oscillation of the spectral channels

We first show that the kth Hankel filter has exactly $k-1$ sign changes, and then relate this oscillation count to representations by positive real decays.

For

$$Z_{st} = \int_0^1 (1-\lambda)^2 \lambda^{s+t-2}\,\mathrm{d}\lambda,$$

let ϕ_k denote the kth eigenvector when the eigenvalues are ordered in decreasing order. Then ϕ_k has exactly $k-1$ sign changes, where zero entries are omitted when counting.

Choose increasing row indices $a_1 < \cdots < a_r$ and column indices $b_1 < \cdots < b_r$. The corresponding minor satisfies

$$\det[Z_{a_i b_j}]_{i,j=1}^{r} = \frac{1}{r!}\int_{(0,1)^r} \det[\lambda_j^{a_i-1}]_{i,j}\det[\lambda_j^{b_i-1}]_{i,j}\prod_{j=1}^{r}(1-\lambda_j)^2\,\mathrm{d}\lambda_1\cdots\mathrm{d}\lambda_r.$$

On $0 < \lambda_1 < \cdots < \lambda_r < 1$, both generalized Vandermonde determinants are positive. Their product is unchanged by a simultaneous permutation of the integration variables and is positive away from sets of measure zero. Hence every minor of Z is positive, so Z is strictly totally positive. The Gantmacher–Krein oscillation theorem (Gantmacher & Krein, 2002) then gives simple positive eigenvalues and exactly $k-1$ sign changes in ϕ_k.

Now consider distinct $\rho_1, \ldots, \rho_m \in (0,1)$ and a nonzero sum

$$f(t) = \sum_{i=1}^{m} c_i \rho_i^t.$$

Such a sum has at most $m-1$ distinct zeros on any real interval and therefore at most $m-1$ sign changes on any ordered sample grid. To see this, first combine repeated poles and remove zero coefficients. For $m > 1$, divide by the positive function ρ_m^t. The derivative of the resulting function is a sum of at most $m-1$ exponentials with distinct rates. Induction and Rolle's theorem then bound the number of zeros of the original sum by $m-1$.

An exact representation of the Hankel filter ϕ_k by positive real decays therefore requires at least k distinct positive poles. A damped rotation is not subject to this sign-change count, since one rotating mode can change sign repeatedly across the horizon. This argument concerns exact representation. Approximation errors at finite mode budgets are measured in Table 28. Table 28 compares the corresponding positive-real and rotating-mode fits empirically.

B.3 Hankel projection error

For $n = s + t - 2$, direct integration gives

$$Z_{st} = \frac{1}{n+1} - \frac{2}{n+2} + \frac{1}{n+3} = \frac{2}{(s+t-1)(s+t)(s+t+1)}. \tag{10}$$

By definition, for $\lambda \sim \mathrm{Unif}[0,1]$,

$$Z = \mathbb{E}_\lambda\big[\mu(\lambda)\mu(\lambda)^\top\big].$$

Among rank-K orthogonal projectors $\mathcal{P}$, the projector

$$\mathcal{P}_K = \sum_{k=1}^{K} \phi_k \phi_k^\top$$

minimizes

$$\mathbb{E}_\lambda\,\|\mu(\lambda) - \mathcal{P}\mu(\lambda)\|_2^2,$$

with minimum error $\sum_{k>K}\sigma_k$. Indeed,

$$\begin{aligned}\mathbb{E}_\lambda\|(I-\mathcal{P})\mu(\lambda)\|_2^2 &= \mathrm{tr}((I-\mathcal{P})Z)\\ &= \mathrm{tr}(Z) - \sum_j \sigma_j\,\phi_j^\top \mathcal{P}\phi_j.\end{aligned}$$

For a rank-K orthogonal projector,

$$0 \le \phi_j^\top \mathcal{P} \phi_j \le 1, \qquad \sum_j \phi_j^\top \mathcal{P} \phi_j = \mathrm{tr}(\mathcal{P}) = K.$$

Since the eigenvalues are ordered as $\sigma_1 \ge \sigma_2 \ge \cdots$,

$$\sum_j \sigma_j \, \phi_j^\top \mathcal{P} \phi_j \le \sum_{j=1}^{K} \sigma_j,$$

with equality for $\mathcal{P} = \mathcal{P}_K$. The resulting projection error is therefore the eigenvalue tail $\sum_{k>K} \sigma_k$.

For a symmetric SSM with eigenvalues $0 \le \lambda_j < 1$, write

$$A = U \Lambda U^\top, \qquad \Gamma_j = (CU)_{:j} (U^\top B)_{j:}.$$

Its temporal kernel is

$$G(\tau) = \sum_j \lambda_j^\tau \Gamma_j.$$

Over the length-L horizon, projecting

$$\nu(\lambda) = [1, \lambda, \ldots, \lambda^{L-1}]^\top$$

onto the first K Hankel filters gives

$$M_k = \sum_j \langle \nu(\lambda_j), \phi_k \rangle \Gamma_j, \qquad G(\tau) \approx \sum_{k=1}^{K} M_k \phi_k(\tau).$$

Because $\mu(\lambda) = (1-\lambda)\nu(\lambda)$,

$$(I - \mathcal{P}_K)\nu(\lambda) = \frac{1}{1-\lambda}(I - \mathcal{P}_K)\mu(\lambda), \qquad \lambda < 1.$$

The projection optimality is stated for the $(1-\lambda)$-weighted family μ. The factor $1/(1-\lambda)$ in the unweighted response remains part of its error bound. The direct term $Du(t)$ is kept separately.

B.4 Expanding the selective recurrence

For a time-varying SSM with zero initial state and readout $y(t) = C_t h(t)$,

$$h(t) = A_t h(t-1) + B_t u(t), \qquad G(t,s) = C_t (A_t A_{t-1} \cdots A_{s+1}) B_s. \tag{11}$$

The empty product at $s = t$ is the identity. We denote the mode transition product by

$$\Pi_{k,i}(t,s) = A_{k,i}(t) A_{k,i}(t-1) \cdots A_{k,i}(s+1), \qquad \Pi_{k,i}(t,t) = I_2.$$

For each ESSH mode,

$$A_{k,i}(t) = \rho_{k,i} \mathrm{e}^{-\Delta_{k,i}(t)} R(\theta_{k,i}),$$

so the scalar decay and fixed planar rotation commute across positions and give the structured transition product in Equation 6.

Fix the realized clock and gate sequences, and define

$$q_{k,i}(t) = \sum_{\ell=1}^{t} \Delta_{k,i}(\ell).$$

Unrolling the mode state from zero gives

$$H_{k,i}(t) = \sum_{s=1}^{t} \Pi_{k,i}(t,s) \eta^{\mathrm{w}}_{k,i}(s) e_1 v_k(s)^\top.$$

For $\tau = t - s$,

$$\begin{aligned} \kappa_{k,i}^\top \Pi_{k,i}(t,s) e_1 &= \rho_{k,i}^\tau \mathrm{e}^{-[q_{k,i}(t) - q_{k,i}(s)]} \kappa_{k,i}^\top R(\theta_{k,i}\tau) e_1 \\ &= \mathrm{e}^{-[q_{k,i}(t) - q_{k,i}(s)]} \psi_{k,i}(\tau), \end{aligned}$$

where

$$\psi_{k,i}(\tau) = \rho_{k,i}^{\tau}\big(\kappa_{k,i}^{\mathrm{c}}\cos(\theta_{k,i}\tau) - \kappa_{k,i}^{\mathrm{s}}\sin(\theta_{k,i}\tau)\big).$$

Therefore,

$$z_k^{\mathrm{core}}(t) = \sum_{s=1}^{t}\sum_{i}\eta_{k,i}^{\mathrm{r}}(t)\eta_{k,i}^{\mathrm{w}}(s)\mathrm{e}^{-[q_{k,i}(t)-q_{k,i}(s)]}\psi_{k,i}(t-s)v_k(s). \tag{12}$$

For fixed realized controls, each source-to-target coefficient therefore factors into a source write gate, path-dependent retention, a lag-dependent damped-mode response, and a target read gate. The resulting map from channel values to outputs is linear under these fixed controls; the complete mixer remains nonlinear because the values, gates, and clocks depend on the mixer input.

Including the zero-lag correction. Equation 24 changes only the coefficient of the current channel value. The implemented output is

$$\begin{aligned} z_k(t) = &\sum_{s=1}^{t-1}\sum_{i=1}^{m}\eta_{k,i}^{\mathrm{r}}(t)\eta_{k,i}^{\mathrm{w}}(s)\mathrm{e}^{-[q_{k,i}(t)-q_{k,i}(s)]}\psi_{k,i}(t-s)v_k(s) \\ &+\left[\sum_{i=1}^{m}(1-a_{k,i})\eta_{k,i}^{\mathrm{r}}(t)\eta_{k,i}^{\mathrm{w}}(t)\kappa_{k,i}^{\mathrm{c}}\right]v_k(t). \end{aligned} \tag{13}$$

At $s = t$, $\Pi_{k,i}(t,t) = I_2$ and $\kappa_{k,i}^{\top}e_1 = \kappa_{k,i}^{\mathrm{c}}$, so the correction replaces only the current-input coefficient by its $(1-a_{k,i})$ multiple. In particular, $\Delta_{k,i}(t)$ acts on the state propagated into position t, not on the value injected at that position.

Effect of an intervening clock. For $s < \ell \le t$, Equation 6 gives

$$\frac{\partial\Pi_{k,i}(t,s)}{\partial\Delta_{k,i}(\ell)} = -\Pi_{k,i}(t,s), \qquad \Pi_{k,i}(t,s)|_{\Delta_{k,i}(\ell)\leftarrow\Delta_{k,i}(\ell)+\delta} = \mathrm{e}^{-\delta}\Pi_{k,i}(t,s), \quad \delta \ge 0. \tag{14}$$

The derivative is zero for $\ell \notin (s,t]$. Thus increasing an intervening clock multiplicatively reduces the contribution propagated from position s to position t without changing its rotation. This derivative treats the realized values and other controls as fixed; it is not the full derivative with respect to the mixer input.

Selective transition versus tokenwise output gating. A three-position example isolates this distinction. Consider one scalar-valued channel with one active mode, $\theta = 0$, $\kappa^{\mathrm{c}} = 1$, $\kappa^{\mathrm{s}} = 0$, and both gates equal to one. Let

$$v(1) \in \{-1,+1\}, \qquad v(2) = v(3) = 0.$$

At position two, vary an input coordinate ignored by the value projection but used by the clock projection, producing $\Delta(2) = \delta_+$ or δ_- with $\delta_+ \ne \delta_-$. Choose the two position-two inputs with equal norms so that normalization does not alter the construction, and keep the position-three mixer input fixed. Then

$$z(3; v(1), \delta_\pm) = \rho^2\mathrm{e}^{-\Delta(3)}\mathrm{e}^{-\delta_\pm}v(1).$$

Because $v(3) = 0$, the zero-lag correction contributes nothing at position three. The mixed finite difference between the earlier value and the intervening clock is therefore

$$\begin{aligned} &z(3;+1,\delta_+) - z(3;-1,\delta_+) - z(3;+1,\delta_-) + z(3;-1,\delta_-) \\ &\quad = 2\rho^2\mathrm{e}^{-\Delta(3)}\big(\mathrm{e}^{-\delta_+} - \mathrm{e}^{-\delta_-}\big) \ne 0. \end{aligned} \tag{15}$$

For the output-gated fixed-filter mixer in Equation 3, the gate at position three is unchanged when its mixer input is fixed. With that gate fixed, the output is additive in the contributions from different source positions, so the same mixed finite difference is zero. The selective transition therefore permits an intervening token to multiplicatively change the later contribution of an earlier value; task-level effects are evaluated in Table 27.

B.5 State bounds and temporal retention

Fix the model parameters and a realized sequence of values and controls. If two mode trajectories differ only in their state at position s, then

$$H_{k,i}(t) - H'_{k,i}(t) = \Pi_{k,i}(t,s)\big(H_{k,i}(s) - H'_{k,i}(s)\big).$$

Because planar rotation preserves the Frobenius norm,

$$\begin{aligned}\|H_{k,i}(t) - H'_{k,i}(t)\|_F &= \rho_{k,i}^{t-s}\mathrm{e}^{-[q_{k,i}(t)-q_{k,i}(s)]}\|H_{k,i}(s) - H'_{k,i}(s)\|_F \\ &\le \rho_{k,i}^{t-s}\|H_{k,i}(s) - H'_{k,i}(s)\|_F.\end{aligned} \tag{16}$$

Thus, for fixed realized controls, an existing mode-state difference is retained from s to t by the factor

$$\rho_{k,i}^{t-s}\mathrm{e}^{-[q_{k,i}(t)-q_{k,i}(s)]}.$$

This factor is also the operator norm of the direct state-to-state map under the Frobenius norm. It does not include the dependence of the realized controls on the mixer input.

Bounded values and states. If $\|u(t)\|_2$ is bounded over the positions under consideration, Equation 23 gives

$$\|v_k(t)\|_2 \le |1-\beta_k|\|W_k^{\mathrm{in}}\|_2\|u(t)\|_2 + |\beta_k|\sqrt{P}.$$

The absolute values are required because β_k is unconstrained. Let $V_{k,\max}$ be any finite bound on the resulting channel values. Since $\|A_{k,i}(t)\|_2 \le \rho_{k,i}$ and $\eta^{\mathrm{w}}_{k,i}(t) \le 2$,

$$\|H_{k,i}(t)\|_F \le \rho_{k,i}\|H_{k,i}(t-1)\|_F + 2V_{k,\max}.$$

Starting from $H_{k,i}(0) = 0$ therefore gives

$$\|H_{k,i}(t)\|_F \le 2V_{k,\max}\sum_{s=1}^{t}\rho_{k,i}^{t-s} = 2V_{k,\max}\frac{1-\rho_{k,i}^{t}}{1-\rho_{k,i}}. \tag{17}$$

The corrected output. Substituting the state update into Equation 24 separates the propagated state from the corrected current injection:

$$z_k(t) = \sum_i \eta^{\mathrm{r}}_{k,i}(t)\left[\big(A_{k,i}(t)H_{k,i}(t-1)\big)^{\top}\kappa_{k,i} + (1-a_{k,i})\eta^{\mathrm{w}}_{k,i}(t)\kappa^{\mathrm{c}}_{k,i}v_k(t)\right].$$

Using the triangle inequality, the gate bounds, and Equation 17 at $t-1$ gives

$$\|z_k(t)\|_2 \le 4V_{k,\max}\sum_i\left[\|\kappa_{k,i}\|_2\frac{\rho_{k,i}(1-\rho_{k,i}^{t-1})}{1-\rho_{k,i}} + |1-a_{k,i}||\kappa^{\mathrm{c}}_{k,i}|\right]. \tag{18}$$

For fixed $V_{k,\max}$ and finite parameters, this bound is uniform in sequence length. Its constant depends on the modal decay and readout coefficients.

B.6 Output change under channel truncation

Fix the input and parameters of one mixer sublayer, and let

$$y_K(t) = K^{-1/2}\sum_{k=1}^{K}W_k^{\mathrm{out}}z_k(t)$$

denote its output when the first K channels are retained, including the zero-lag correction. Define

$$S_K(t) = \sum_{k=1}^{K}W_k^{\mathrm{out}}z_k(t).$$

The difference between the full mixer and the K-channel mixer is exactly

$$\begin{aligned}y_{\bar{K}}(t) - y_K(t) = \frac{1}{\sqrt{\bar{K}}}\sum_{k>K}W_k^{\mathrm{out}}z_k(t) \\ + \big(\bar{K}^{-1/2} - K^{-1/2}\big)S_K(t).\end{aligned} \tag{19}$$

The first term contains the discarded channels, while the second accounts for the tier-dependent output normalization.

To bound the corrected channel output, define

$$\widetilde{G}_k(\tau) = \begin{cases} \sum_i |1 - a_{k,i}|\,|\kappa^{\mathrm{c}}_{k,i}|, & \tau = 0, \\ \sum_i |\psi_{k,i}(\tau)|, & \tau > 0. \end{cases}$$

Equation 13, together with $\eta^{\mathrm{w}}_{k,i}, \eta^{\mathrm{r}}_{k,i} < 2$ and $\mathrm{e}^{-[q_{k,i}(t) - q_{k,i}(s)]} \le 1$, gives

$$\|z_k(t)\|_2 \le 4 \sum_{s=1}^{t} \widetilde{G}_k(t-s) \|v_k(s)\|_2.$$

Applying the triangle inequality and $\|Wz\|_2 \le \|W\|_2 \|z\|_2$ to Equation 19 yields

$$\begin{aligned} \|y_{\bar{K}}(t) - y_K(t)\|_2 \le \frac{4}{\sqrt{\bar{K}}} \sum_{k>K} \|W^{\mathrm{out}}_k\|_2 \sum_{s=1}^{t} \widetilde{G}_k(t-s) \|v_k(s)\|_2 \\ + \left| \bar{K}^{-1/2} - K^{-1/2} \right| \|S_K(t)\|_2. \end{aligned} \tag{20}$$

Equation 20 separates the two direct effects of channel truncation at a fixed mixer input: removing higher-indexed channel contributions and changing the tier normalization. Which retained configurations remain predictive is determined by the learned channel organization and capacity-mixed training.

B.7 Mode-fit approximation and initialization

Define the relative mode-fit error

$$\varepsilon^{(m)}_k = \frac{\|\phi_k - \widehat{\phi}^{(m)}_k\|_2}{\|\phi_k\|_2}.$$

With $m = 8$, the first eight channels retained by the smallest deployment tier have relative fit error at most 0.036. Higher-index filters are more oscillatory, and their errors are weighted by smaller eigenvalues in Equation 21. At $L = 2048$, $\sigma_9/\sigma_1 = 2.29 \times 10^{-6}$ and the tail beyond channel eight is $3.31 \times 10^{-6} \operatorname{tr} Z$; the projection-plus-fit error is below $4 \times 10^{-6} \operatorname{tr} Z$ at every deployment tier. These statistics concern the fitted weighted filter family. The modal parameters remain trainable.

Combining channel truncation with finite mode fits. The Hankel projection in Appendix B.3 and the finite-mode approximation contribute separate terms to the reconstruction error of the weighted family μ. Using the original Hankel projection coefficients with the fitted filters, define

$$\begin{aligned} \mathcal{E}^{(m)}_K &= \mathbb{E}_{\lambda \sim \mathrm{Unif}[0,1]} \left\| \mu(\lambda) - \sum_{k=1}^{K} \langle \mu(\lambda), \phi_k \rangle \widehat{\phi}^{(m)}_k \right\|_2^2 \\ &= \sum_{j>K} \sigma_j + \sum_{k=1}^{K} \sigma_k \big(\varepsilon^{(m)}_k\big)^2. \end{aligned} \tag{21}$$

The first term is the Hankel projection tail from omitting channels $k > K$, while the second is the error from representing each retained filter with m modes.

To verify the second line, expand $\mu(\lambda)$ in the orthonormal eigenbasis of Z. The residual is

$$\sum_{j>K} \langle \mu, \phi_j \rangle \phi_j + \sum_{k=1}^{K} \langle \mu, \phi_k \rangle \big(\phi_k - \widehat{\phi}^{(m)}_k\big).$$

For distinct eigenvectors,

$$\mathbb{E}_\lambda \left[\langle \mu, \phi_j \rangle \langle \mu, \phi_k \rangle \right] = \phi_j^\top Z \phi_k = \sigma_k \mathbf{1}[j = k],$$

so the cross terms vanish after integration. Since $\|\phi_k\|_2 = 1$, the remaining diagonal terms give Equation 21. The fitted filters themselves need not be orthogonal.

Implemented initialization. The fitted response is the modal sum before input-dependent controls. The implemented initialization also applies the initial clock and zero-lag correction from Appendix C.3. Its response is therefore evaluated using the complete recurrence in Equations 9 and 24.

For the unmodulated fitted response, the finite-horizon filter error also gives

$$\left\| \sum_{\tau=0}^{t-1} \big(\phi_k(\tau) - \widehat{\phi}_k^{(m)}(\tau)\big) v_k(t-\tau) \right\|_2 \le \varepsilon_k^{(m)} \left(\sum_{s=1}^{t} \|v_k(s)\|_2^2 \right)^{1/2}, \qquad t \le L. \tag{22}$$

This follows from the triangle inequality and Cauchy–Schwarz over sequence positions.

These bounds characterize the initialization. During training, the modal parameters, clocks, and gates are learned, and the resulting dynamics are described by the selective recurrence in Equation 12.

C Mixer Parameterization and Cost

This section specifies the spectral initialization, channel-value transformation, and zero-lag correction that complete the selective recurrence in Equation 5. It then gives the parameter, arithmetic, and recurrent-state costs of each capacity tier.

C.1 Initialization from Hankel channels

We compute the first $\bar{K} = 32$ unit eigenvectors of the length-2048 Hankel matrix and fit each with a small bank of damped rotation modes. Filter-fit diagnostics use the first 24 eigenvectors resolved by the double-precision conditioning check. All 32 fitted modal parameter sets are trainable initializations; the weighted approximation error is quantified in Appendix B.7.

C.1.1 The matrix-pencil fit

For each length-2048 channel, the matrix-pencil procedure uses pencil parameter 512 and a rank-m truncated SVD to estimate m poles. Each pole is stored as one damped mode, so conjugate poles occupy separate modes. After clipping the pole magnitudes to stable decays, complex readout coefficients are obtained by least squares. The resulting $(\rho, \theta, \kappa^{\mathrm{c}}, \kappa^{\mathrm{s}})$ tables initialize every spectral layer.

With $m = 8$, the first eight channels, which are retained by T_1, have maximum relative fitting error 0.036. Fit error increases for higher-index, more oscillatory channels, while all modal parameters remain trainable after initialization. Appendices L.2 and L.4 evaluate fitting fidelity and trained-model behavior separately.

C.2 Projections, gates, and clock

The spectral block receives the RMS-normalized mixer input $u(t)$ (Zhang & Sennrich, 2019). Each channel first computes $v_k^0(t) = W_k^{\mathrm{in}} u(t)$ and then applies the learned channel-local blend

$$v_k(t) = (1 - \beta_k) v_k^0(t) + \beta_k \frac{v_k^0(t)}{\sqrt{P^{-1}\|v_k^0(t)\|_2^2 + \epsilon_{\mathrm{norm}}}}, \tag{23}$$

where β_k is a learned scalar initialized at zero and $\epsilon_{\mathrm{norm}} > 0$ stabilizes the denominator. The transformation therefore starts as the identity. Its normalization statistic uses only the P coordinates of channel k, so removing other channels does not change a retained channel value for the same mixer input.

For channel k and mode i, three affine projections of $u(t)$ produce the write gate, read gate, and clock increment:

$$\begin{aligned} \eta_{k,i}^{\mathrm{w}}(t) &= 2\,\mathrm{sigmoid}\big((w_{k,i}^{\mathrm{w}})^\top u(t) + b_{k,i}^{\mathrm{w}}\big), \\ \eta_{k,i}^{\mathrm{r}}(t) &= 2\,\mathrm{sigmoid}\big((w_{k,i}^{\mathrm{r}})^\top u(t) + b_{k,i}^{\mathrm{r}}\big), \\ \Delta_{k,i}(t) &= \mathrm{softplus}\big((w_{k,i}^{\Delta})^\top u(t) + b_{k,i}^{\Delta}\big). \end{aligned}$$

All three projection vectors lie in $\mathbb{R}^d$, and their biases are scalars. Gate weights and biases are initialized at zero, so both gates start at one. Clock weights are initialized at zero with bias -3, giving $\Delta_0 = \mathrm{softplus}(-3) \approx 0.0486$. This corresponds to a clock-only e-folding scale of approximately 20.6 positions at initialization. Because the clock parameters are learned, this additional attenuation can increase or decrease during training.

C.3 Stable decay and zero-lag correction

The implementation stores $\tilde{\rho}$ and parameterizes

$$\gamma = \mathrm{softplus}(\tilde{\rho}), \qquad \rho = \exp(-\gamma).$$

Hence

$$\|A_{k,i}(t)\|_2 = \exp\big(-\gamma_{k,i} - \Delta_{k,i}(t)\big) < 1.$$

Each mode transition is therefore contractive, while the learned $\tilde{\rho}_{k,i}$ allows its intrinsic timescale to increase or decrease during training.

Each mode also has a learned zero-lag correction coefficient $a_{k,i}$, initialized at $1/2$. The corrected channel output is

$$z_k(t) = z_k^{\mathrm{core}}(t) - \sum_i a_{k,i}\eta^{\mathrm{r}}_{k,i}(t)\eta^{\mathrm{w}}_{k,i}(t)\kappa^{\mathrm{c}}_{k,i}v_k(t). \tag{24}$$

The correction changes only the lag-zero coefficient and requires no additional recurrent state.

To see the equivalent recurrence, suppress the mode index, let $b_t = \eta^{\mathrm{w}}(t)e_1 v(t)^\top$, and define $J_t = H_t - ab_t$. With $b_0 = 0$,

$$J_t = A_t J_{t-1} + (1-a)b_t + aA_t b_{t-1},$$

and

$$\eta^{\mathrm{r}}(t)J_t^\top\kappa = \eta^{\mathrm{r}}(t)H_t^\top\kappa - a\eta^{\mathrm{r}}(t)b_t^\top\kappa.$$

Thus the corrected readout can be computed from the existing mode state and current channel value. At the initialization $a = 1/2$, the equivalent recurrence assigns equal coefficients to the current injection and the propagated previous injection.

At initialization the gates equal one and the clock is $\Delta_0 = \mathrm{softplus}(-3)$. Let $\psi^0_{k,i}$ denote the fitted modal response from Equation 12. The implemented initial response from channel values to channel output is

$$\widehat{\phi}_{k,\mathrm{init}}(\tau) = \mathrm{e}^{-\Delta_0\tau}\sum_i \psi^0_{k,i}(\tau) - \mathbf{1}[\tau = 0]\sum_i a^0_{k,i}\kappa^{\mathrm{c},0}_{k,i}.$$

The first term incorporates the initial clock attenuation, while the second applies the learned form of the lag-zero correction. This is the implemented initialization analyzed in Appendix B.7.

C.4 Channel aggregation and the complete computation

At tier T, the mixer computes the first K_T channel values and their channel-local blends, evaluates the corresponding $K_T m$ gate and clock rows, updates the retained mode states, and applies Equation 24. The corrected channel outputs are concatenated and projected through the first $K_T P$ columns of W^{out}, with scale $K_T^{-1/2}$. The resulting width-d output is added to the residual stream.

No data-dependent normalization statistic is shared across spectral channels. For the same mixer input, truncating later channels therefore leaves the internal computation of every retained channel unchanged.

With the value blend disabled, the linear input–output map of channel k factors as

$$W^{\mathrm{out}}_k W^{\mathrm{in}}_k,$$

and has rank at most P. ESSH therefore uses a parameter-efficient factorized channel map in place of the dense M_k in the reference spectral construction of Equation 2.

C.5 PARAMETER AND FLOP ACCOUNTING

Table 4 separates the projection parameters from the smaller modal tables. At full capacity, the value and output projections contain $2d^2$ parameters, and the three selective projections contain $3\bar{K}md$ parameters before biases. A SwiGLU block contains $3d\,d_{\text{ff}}$ projection parameters. At the 370M configuration, the feed-forward layer accounts for approximately 84% of the parameters in a spectral block. This separation motivates reducing feed-forward width more aggressively than spectral-channel count in the two-rate capacity map.

We count one multiply-accumulate as two FLOPs. At tier T, the value/output and selective projections require

$$2dK_TP + 3dK_Tm$$

MACs per token, while the feed-forward projections require

$$3d\,n_{\text{ff},T}$$

MACs. The recurrent core requires $O(K_TmP)$ tokenwise work and $2K_TmP$ recurrent-state entries per spectral layer. Attention remains shared across tiers and contributes its own projection cost and window KV cache. Table 5 summarizes the corresponding full-capacity costs.

For Mamba-2, d_{inner} denotes its expanded mixer width and N_{state} its state dimension per value coordinate; these are distinct from the generic SSM dimensions in Table 2. The asymptotic expressions omit lower-order buffers and device-dependent effects such as memory traffic and kernel-launch overhead, which are measured directly in the systems evaluation.

Exact tier-dependent parameter count. For the stated bias-free value/output and SwiGLU projections, each retained spectral channel contributes

$$2dP + 3md + 8m + 1$$

parameters: value/output weights, three selective-projection weights and biases, five learned modal scalars per mode, and one value-blend coefficient. With N_{layers} blocks and attention-layer set $\mathcal{A}$,

$$\begin{aligned}\text{Params}(T) = \;& \text{Params}_{\text{fixed}} \\ & + (N_{\text{layers}} - |\mathcal{A}|)K_T(2dP + 3md + 8m + 1) \\ & + 3N_{\text{layers}}d\,n_{\text{ff},T}.\end{aligned} \tag{25}$$

The fixed term contains embeddings, attention projections, and normalization parameters. The recurrent state contains

$$2(N_{\text{layers}} - |\mathcal{A}|)K_TmP$$

entries and is independent of $n_{\text{ff},T}$.

Thus the two elastic axes control different deployment costs: feed-forward width primarily changes parameter count and feed-forward computation, whereas spectral-channel count changes mixer computation and recurrent-state storage in addition to channel parameters. The shared window KV cache remains fixed across tiers.

Table 4: Parameters per block for the two language-model configurations, with $\bar{K} = 32$, $m = 8$, and $P = d/\bar{K}$. Whole-model counts include tied embeddings.

Component	Parameters	1.5B (d=2048, d_{ff}=5632)	370M (d=896, d_{ff}=4608)
value in / out projections	$2d^2$	8,388,608	1,605,632
write, read gates	$2(\bar{K}m\,d + \bar{K}m)$	1,049,088	459,264
clock	$\bar{K}m\,d + \bar{K}m$	524,544	229,632
spectral tables $(\rho, \theta, \kappa^{\text{c}}, \kappa^{\text{s}})$	$4\bar{K}m$	1,024	1,024
zero-lag coefficient a, value blend β	$\bar{K}m + \bar{K}$	288	288
SwiGLU feed-forward (Shazeer, 2020)	$3d\,d_{\text{ff}}$	34,603,008	12,386,304
sliding-window attention (per SWA block)	$4d^2$	16,777,216	3,211,264
two RMSNorms	$2d$	4,096	1,792
whole model, full capacity		1,531,089,696	370,866,144
whole model, exported T_4		827,570,754	185,254,998

Table 5: Per-token arithmetic and per-sequence persistent state at full capacity. Compact-tier spectral and feed-forward costs follow by replacing $\bar{K}$ and d_{ff} with K_T and $n_{\text{ff},T}$, while attention remains shared. The Mamba-2 row assumes a shared B/C group and $N_{\text{state}} \leq d_{\text{inner}}$, as in the timed configuration (Dao & Gu, 2024). One multiply-accumulate counts as two FLOPs.

Block	projection MACs + scan cost	decode cost per token	persistent state
Spectral mixer	$2d^2 + 3\bar{K}md + O(Q\bar{K}(P+m) + \bar{K}mP)$	$O(d^2 + \bar{K}md + md)$	$2\bar{K}mP = 2md$
Window attention	$4d^2 + O(wd)$	$O(d^2 + wd)$	$2wd$
SwiGLU feed-forward	$3d\,d_{\text{ff}}$	$O(d\,d_{\text{ff}})$	0
Mamba-2 block	$O(d\,d_{\text{inner}}) + O((Q + N_{\text{state}})d_{\text{inner}})$	$O(d\,d_{\text{inner}} + N_{\text{state}}d_{\text{inner}})$	$N_{\text{state}}d_{\text{inner}} + O(d_{\text{inner}})$
Full attention	$4d^2 + O(Ld)$	$O(d^2 + Ld)$	$2Ld$

D Elastic Training

This section specifies the nested tensor slices, capacity-mixed sampling, gradient allocation, full-tier distillation, and standalone export used to train the elastic family.

Figure 5 summarizes the two-rate capacity map and the capacity-mixed training rule.

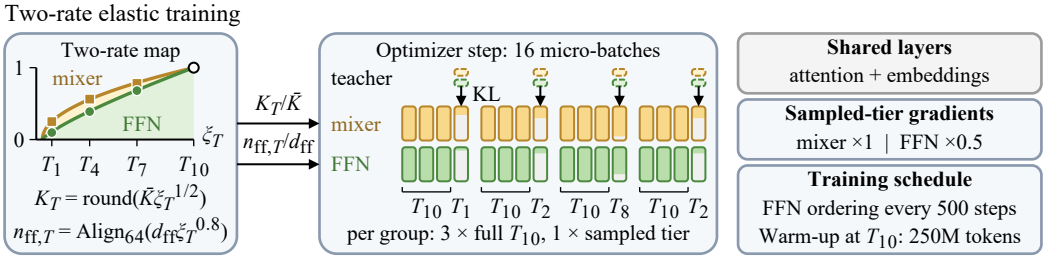

Figure 5: Two-rate elastic training. The two-rate map reduces spectral-channel count and feed-forward width at different rates; $\text{Align}_{64}(x) = 64\max\{1, \lfloor x/64 \rceil\}$. The optimizer-step diagram shows joint training of full and sampled capacities with full-tier distillation. The right-hand boxes identify the shared attention and embeddings, reduced-capacity gradient scales, and the capacity warm-up and FFN-ordering schedule.

D.1 The two-rate capacity map

Equation 7 maps a budget variable to a retained spectral-channel count and feed-forward width. Attention, residual width, embeddings, and normalization dimensions remain fixed across tiers. The exponents $1/2$ and 0.8 are empirical recipe choices evaluated in Appendix M.1.

At the 370M configuration, for example, T_1 retains eight of 32 spectral channels and 448 of 4608 feed-forward units. The map therefore preserves a larger fraction of recurrent channels while concentrating most parameter reduction in the feed-forward layers. Attention dimensions and window size remain shared across tiers.

Table 6 lists the ten deployment tiers, whose budget fractions are

$$(\xi_{T_1}, \ldots, \xi_{T_{10}}) = \frac{1}{32}(2, 4, 7, 10, 13, 17, 20, 24, 28, 32).$$

The pretraining sampler uses the separate ten training capacities

$$\{1/16,\ 3/32,\ 1/8,\ 5/32,\ 3/16,\ 1/4,\ 3/8,\ 1/2,\ 3/4,\ 1\}.$$

The ten training capacities place six budgets at $\xi \leq 1/4$. Deployment tiers evaluate slices of the same nested tensors. At 370M, T_3–T_7 and T_9 lie between the capacities directly sampled during training; Table 10 evaluates these intermediate exports alongside the directly sampled configurations. Results

under the default map use Table 6; alternative-map ablations keep the budget variables and report their resulting sizes.

Table 6: The ten deployment tiers and measured parameter counts. Channel count and FFN width follow the two-rate map. Fixed embedding parameters cause the exported parameter fractions to differ between the two model scales.

tier	K_T	$n_{\mathrm{ff},T}$, 1.5B	$n_{\mathrm{ff},T}$, 370M	params, 1.5B (B)	params, 370M (M)	modes $K_T m$
T_1	8	576	448	0.474	92.1	64
T_2	11	1024	832	0.575	118.9	88
T_3	15	1664	1344	0.716	154.7	120
T_4	18	2176	1792	0.828	185.3	144
T_5	20	2688	2240	0.931	214.5	160
T_6	23	3392	2752	1.076	248.8	184
T_7	25	3840	3136	1.168	274.3	200
T_8	28	4416	3648	1.291	308.6	224
T_9	30	5056	4096	1.416	337.9	240
T_{10}	32	5632	4608	1.531	370.9	256

D.2 Standalone export, tensor by tensor

At tier T, retain the first $K_T P$ value-projection rows and matching output-projection columns, the first $K_T m$ gate and clock rows and biases, and the first K_T channel blocks of the modal tables, zero-lag coefficients, and channel-local blend coefficients. In each feed-forward layer, retain the first $n_{\mathrm{ff},T}$ rows of the gate and up projections and the matching columns of the down projection.

In exact arithmetic, these slices form a standalone model that computes the same function as the shared model evaluated at tier T. Every retained operation uses the same parameters and operands in both evaluations: channel-local normalization keeps width P unchanged, retained mode states therefore match by induction over sequence positions, attention is shared, and the same FFN units are selected. The equality then propagates through the residual blocks to the final normalization and output head.

Different kernel reduction orders can introduce floating-point differences. Export checks therefore compare the implemented standalone and shared model outputs at the specified numerical precision.

D.3 Importance ordering of the feed-forward units

Because reduced-capacity tiers retain the first $n_{\mathrm{ff},T}$ feed-forward units, their ordering determines which units remain in reduced-capacity models. A full-width SwiGLU block is invariant to a common permutation of the gate and up rows and the corresponding down-projection columns.

Every 500 steps through the first 80% of training, the recipe orders units using a running squared-activation score weighted by the squared norm of the corresponding down-projection column. The same permutation is applied to the gate and up rows, down columns, optimizer moments, and running activation statistics. Distributed workers synchronize the scores before applying the permutation.

The permutation preserves the current full-width function in exact arithmetic while changing which units occupy the reduced-capacity models; subsequent capacity-mixed training adapts the reduced-capacity models. Spectral channels are not reordered.

D.4 Capacity-mixed micro-batches

In the default recipe, all micro-batches use full capacity during the first 250M training tokens. The adaptive sampling variants in Appendix M.2 use the activation schedules specified there. After this warm-up, each optimizer step reserves a fraction $f = 0.75$ of its micro-batches for the full-capacity tier. Each remaining micro-batch independently draws uniformly from the ten training capacities above, including full capacity. This sampling operation is budget dropout. The expected

full-capacity share after warm-up is therefore

$$0.75 + \frac{0.25}{10} = 0.775.$$

For a reduced-capacity micro-batch, each retained feed-forward weight tensor is evaluated as

$$\widehat{W} = \alpha_{\mathrm{ff}} W + (1 - \alpha_{\mathrm{ff}})\,\mathrm{sg}(W), \qquad \alpha_{\mathrm{ff}} = 0.5.$$

Since $\widehat{W} = W$ in the forward pass, this transformation leaves activations unchanged. Under differentiation, the parameter gradient contributed by the reduced-capacity branch is multiplied by α_{ff}, while the gradient with respect to the layer input is unchanged. The scaling applies only to retained feed-forward weights; mixer and attention parameters use their ordinary gradients. Compute savings come from evaluating the smaller retained tensor slices rather than from the gradient scaling itself. Reserved full-capacity micro-batches ensure that every parameter group participates in each optimizer step.

D.5 Distillation from the full model

During pretraining, a reduced-capacity micro-batch uses the full-tier distillation objective of Equation 8:

$$\mathcal{L}_T = \mathrm{CE}(p_T, \mathrm{id}_{t+1}) + 0.5\,\mathrm{KL}\big(\mathrm{sg}[p_{T_{10}}] \,\|\, p_T\big). \tag{26}$$

The teacher is the current full-capacity tier evaluated on the same tokens. Its prediction is stop-gradient, so the distillation term updates the reduced-capacity predictor without backpropagating through the teacher distribution. Each reduced-capacity micro-batch therefore evaluates both the full-capacity teacher and the retained student. Full-capacity micro-batches use the next-token cross-entropy term alone.

D.6 Chunked loss evaluation

Cross-entropy and distillation are sums over token positions and can therefore be evaluated in contiguous position chunks while retaining the complete vocabulary within each chunk. Under bf16 autocast, each chunk computes logits in bf16 and log-softmax and loss accumulation in fp32. Activation checkpointing recomputes the chunk during backward. Weighting chunk losses by their valid-token counts recovers the same full-sequence objective and gradients while reducing peak logit storage. This optional path uses a fixed 4096-token loss chunk when enabled; the main 370M run and shared refinement use full-logit loss evaluation.

D.7 One optimizer step

Algorithm 1 summarizes one step of the default capacity-mixed pretraining recipe. Here N_r is the valid-token count of micro-batch r, N_{tok} is the global valid-token count, g_{acc} is the accumulated parameter gradient, $N_{\mathrm{warm}} = 250\mathrm{M}$, $j_{\max}$ is the final pretraining step, and $\bar{a}^2_{\ell,h}$ is the running squared activation of FFN unit h. Slice returns parameter views tied to Θ, and $\mathrm{PermuteFFN}$ applies the same unit permutation to the FFN parameters, optimizer state, and running activation statistics.

Algorithm 1: Capacity-mixed pretraining

Input: Θ, Ω, micro-batches $\{(x_r, \mathrm{id}_r, N_r)\}_{r=1}^{M}$, reserved full slots $\mathcal{F}$, training capacities $\mathcal{T}$, tokens seen n, step j
Output: updated parameters Θ, optimizer state Ω, and sampled capacities $\{T_r\}_{r=1}^{M}$

$g_{\mathrm{acc}} \leftarrow 0$
$N_{\mathrm{tok}} \leftarrow \mathrm{AllReduceSum}(\sum_{r=1}^{M} N_r)$
for $r = 1, \ldots, M$ **do**
  if $n < N_{\mathrm{warm}}$ *or* $r \in \mathcal{F}$ **then**
    $T_r \leftarrow T_{10}$
  else
    $T_r \sim \mathrm{Sample}(\mathcal{T})$
  if $T_r = T_{10}$ **then**
    $p_r \leftarrow \mathrm{Model}(x_r; \Theta, T_{10})$
    $\ell_r \leftarrow \mathrm{CE}(p_r, \mathrm{id}_r)$
  else
    $p_{\mathrm{full}} \leftarrow \mathrm{sg}[\mathrm{Model}(x_r; \Theta, T_{10})]$
    $\Theta_r \leftarrow \mathrm{Slice}(\Theta, K_{T_r}, n_{\mathrm{ff},T_r})$
    $\widehat{\Theta}_r \leftarrow \Theta_r$
    foreach *retained FFN weight* W *in* Θ_r **do**
      $\widehat{\Theta}_r[W] \leftarrow \alpha_{\mathrm{ff}} W + (1 - \alpha_{\mathrm{ff}})\,\mathrm{sg}(W)$
    $p_r \leftarrow \mathrm{Model}(x_r; \widehat{\Theta}_r, T_r)$
    $\ell_r \leftarrow \mathrm{CE}(p_r, \mathrm{id}_r) + 0.5\,\mathrm{KL}(p_{\mathrm{full}} \| p_r)$
  $g_{\mathrm{acc}} \leftarrow g_{\mathrm{acc}} + (N_r / N_{\mathrm{tok}}) \nabla_{\Theta} \ell_r$
$g_{\mathrm{acc}} \leftarrow \mathrm{AllReduceSum}(g_{\mathrm{acc}})$
$(\Theta, \Omega) \leftarrow \mathrm{AdamW}(\Theta, \Omega, \mathrm{Clip}(g_{\mathrm{acc}}, 1))$
if $j > 0$, $j \bmod 500 = 0$, *and* $j < 0.8 j_{\max}$ **then**
  foreach *FFN layer* ℓ **do**
    $I_{\ell,h} \leftarrow \mathrm{AllReduceMean}(\bar{a}_{\ell,h}^2 \| W_{\ell,:,h}^{\mathrm{down}} \|_2^2)$
    $\pi_\ell \leftarrow \mathrm{argsort}_{\downarrow}(I_\ell)$
    $(\Theta_\ell, \Omega_\ell, \bar{a}_\ell^2) \leftarrow \mathrm{PermuteFFN}(\Theta_\ell, \Omega_\ell, \bar{a}_\ell^2; \pi_\ell)$
return $\Theta, \Omega, \{T_r\}_{r=1}^{M}$

D.8 Standalone export

After capacity-mixed pretraining, shared refinement (SR) continues joint training of the full model and sampled reduced-capacity tiers. It combines their label losses with total-variation distillation from the full prediction:

$$\mathcal{L}_{\mathrm{SR}} = \mathrm{CE}(p_{T_{10}}, \mathrm{id}_{t+1}) + \mathrm{CE}(p_T, \mathrm{id}_{t+1}) + 5\,\mathrm{TV}(\mathrm{sg}[p_{T_{10}}], p_T), \quad \mathrm{TV}(p, q) = \tfrac{1}{2} \sum_{v \in \mathcal{V}} |p(v) - q(v)|.$$

Here $\mathcal{V}$ is the vocabulary. The full model receives gradients from its label loss, while the teacher distribution in the distillation term is held fixed. At 370M, one capacity is sampled uniformly per optimizer step from $\{1/16, 1/8, 1/4, 1/2, 3/4\}$ and is shared by that step's micro-batches. At 1.5B, each step shuffles the nine reduced deployment tiers T_1–T_9 and cycles through that order across its 16 micro-batches. Every micro-batch evaluates the full model and its sampled capacity. SR initializes a fresh AdamW optimizer with betas $(0.9, 0.95)$, gradient clipping at 1.0 and weight decay 0.1 on tensors with at least two dimensions; this includes embeddings and modal parameter tables. The learning rate warms up for 5% of the steps and follows cosine decay toward zero. The retained FFN gradient scale remains $\alpha_{\mathrm{ff}} = 0.5$. Scale-specific settings are listed in Table 55.

Every standalone export is obtained by slicing this final checkpoint as specified in Appendix D.2. Each export stores only its retained tensors and configuration. The main ESSH results use the final model after the complete procedure in Appendix Q.1. The 370M ablation tables also report end-of-pretraining measurements where specified, with the same training stage used across arms within each column.

The export check compares each standalone model with its source shared model at the same tier on two validation batches of 4×2048 tokens. The comparison uses the same precision and kernel

path for the shared model and standalone export. All ten final 1.5B exports match the shared model evaluated at the same tier bitwise on H200, using the same precision and kernel path.

E EXPERIMENTAL PROTOCOLS

This section specifies the data, model configurations, training budgets, evaluation protocols, hardware, and baseline matching used in the experiments. Appendix D.8 defines the final ESSH model and the training stages used in the 370M ablations.

E.1 DATA AND TOKENIZATION

The 370M experiments use FineWeb-Edu with the GPT-2 tokenizer and a 50,304-entry model vocabulary. Main runs process 7.4B tokens, while mechanism and hyperparameter ablations use 3.7B tokens. The 1.5B run processes 100B FineWeb-Edu pretraining tokens using the Llama-3.1 tokenizer and a 128,256-entry vocabulary. All training sequences are packed to length 2048 (Penedo et al., 2024; Radford et al., 2019; Grattafiori et al., 2024).

The token budgets count processed tokens. Data-preparation commands and stream checksums are provided with the code.

At 370M, the perplexity frontier uses the validation file prepared with the 30B-token stream, while retrieval filler, length sweeps and export checks use the file prepared with the 5B-token stream. Local model comparisons use fixed evaluation inputs within each protocol. The 1.5B experiments use the separately tokenized Llama-3.1 stream. Cross-domain comparisons use fixed token windows within each tokenizer group.

E.2 CONFIGURATIONS

Table 7 lists the two language-model configurations. Both use the same selective spectral mixer parameterization and two-rate elastic recipe, while depth, residual width, feed-forward width, attention positions, tokenizer, learning rate, and token budget vary with scale. Complete run-specific launch configurations are included with the experimental artifacts. Distributed training preserves the stated global token batch across worker configurations. During pretraining, raw decay parameters, frequencies and modal readout coefficients are excluded from weight decay, as are embeddings, normalization parameters and biases. The two-dimensional zero-lag coefficient table uses weight decay 0.1. SR uses the optimizer groups defined in Appendix D.8.

E.3 TOKEN BUDGETS AND TRAINING EXPOSURE

The main 370M models and independently trained specialists use 7.4B pretraining tokens. The dense parents in the recovery controls use 3.7B tokens. Mechanism and hyperparameter ablations use 3.7B tokens with learning-rate schedules defined over that budget. The shared final stage in Table 55 brings total ESSH exposure to 7.67B tokens for the main runs and 3.97B for the ablations.

Post-hoc elastic baselines begin from their pretrained parents and use an additional 0.5B recovery tokens. Their corresponding total data exposures are therefore reported separately as 7.90B or 4.20B tokens. Training-token exposure and measured accelerator time are both reported when comparing the cost of producing multiple deployment sizes.

E.4 EVALUATION PROTOCOLS

Figure 6 reports absolute task accuracy and its change relative to T_{10}. Table 15 gives the corresponding scores and model sizes.

The seven-task mean is the unweighted average of LAMBADA accuracy, HellaSwag normalized accuracy, PIQA accuracy, ARC-Easy accuracy, ARC-Challenge normalized accuracy, WinoGrande accuracy, and OpenBookQA accuracy, evaluated zero-shot with lm-evaluation-harness v0.4.9.1 (Sutawika et al., 2025; Paperno et al., 2016; Zellers et al., 2019; Bisk et al., 2020; Clark et al., 2018; Sakaguchi et al., 2020; Mihaylov et al., 2018). Additional evaluations use BoolQ, SIQA,

Table 7: Language-model configurations. Common algorithmic choices are listed alongside scale-specific depth, width, tokenizer, learning rate, and training budget.

	370M	1.5B
blocks / width / feed-forward width	22 / 896 / 4608	28 / 2048 / 5632
channels $\bar{K}$ / modes m / value width P	32 / 8 / 28	32 / 8 / 64
sliding-window layers (zero-based; window 2048, RoPE (Su et al., 2024), head dim. 64)	blocks 12, 17, 21	blocks 15, 22, 27
tokenizer / embedding	GPT-2, tied	Llama-3.1, tied
parameters at full capacity	370.9M	1.531B
pretraining tokens	7.4B main / 3.7B ablations	100B
tokens per step	262,144	524,288
peak learning rate, schedule	1.2×10^{-3}, warm-up 1%, cosine to 10%	1.0×10^{-3}, same
optimizer	AdamW (Loshchilov & Hutter, 2019) $(0.9, 0.95)$, wd 0.1, clip 1.0, bf16	same
full share f / slice scale α_{ff} / KL coefficient	0.75 / 0.5 / 0.5	same
training budget fractions	$1/16$, $3/32$, $1/8$, $5/32$, $3/16$, $1/4$, $3/8$, $1/2$, $3/4$, 1	same
validation / checkpoint	every 500 steps / 250 (main run)	every 1000 / 200

SciQ, COPA, RACE, and LogiQA (Clark et al., 2019; Sap et al., 2019; Welbl et al., 2017; Roemmele et al., 2011; Lai et al., 2017; Liu et al., 2020). Scores are reported as fractions unless a table explicitly uses percentages. The published 1.5B rows are transcribed from Table 1 of Lahoti et al. (2026), which labels OpenBookQA as accuracy; ESSH uses the same metric.

All 370M checkpoints in the main frontier comparison are evaluated on the same held-out shard with the same evaluator: 16 batches of eight length-2048 sequences under bf16 autocast. Repeated evaluations are used only as numerical checks rather than as independent experimental replicates. The separately tokenized 1.5B validation stream is not used for direct 370M cross-model PPL comparisons.

Needle retrieval. The NIAH evaluation inserts a six-digit key into a 2048-token filler context at five relative depths. A key pair contains a planted key and its counterfactual alternative; margin win is the fraction of trials in which the planted key receives a higher conditional score. The NIAH tables use the counterfactual condition, which inserts the alternative key and treats it as the expected answer. The 1280 trials comprise 256 key pairs evaluated at each depth. The primary retrieval metric is teacher-forced full-key accuracy: every key token must be predicted correctly while conditioning on the correct preceding key tokens. First-token top-1 accuracy and candidate-margin win rate are reported separately where used.

Confidence intervals resample the 256 key pairs with replacement while keeping the five depths of each pair together; reported 95% intervals use 2000 bootstrap replicates.

Cross-domain language modeling. Cross-domain evaluation uses fixed text slices from PG-19, the Pile arXiv subset, GitHub code from CodeParrot, CC-News, Proof-Pile, and WikiText-103 (Rae et al., 2020; Gao et al., 2020; CodeParrot, 2022; Hamborg et al., 2017; Azerbayev et al., 2022; Merity et al., 2017). Documents are concatenated with newline separators and partitioned into non-overlapping windows of 2049 tokens. Each window supplies 2048 input tokens and 2048 next-token targets. Perplexity is the exponential of the mean target-token negative log-likelihood. All models within a tokenizer group use the same token windows.

E.5 Hardware, precision, and the timing harness

Training uses bf16 autocast with fp32 master weights and fp32 scan accumulation. Unless a table specifies another device, systems measurements use one A100-SXM4-80GB. ESSH uses bf16

weights and window KV cache with fp32 recurrent state; baselines use the cache and state dtypes of their evaluated implementations. Memory totals are computed from allocated tensors.

Decode latency measures one autoregressive inference step with state and cache sized for a 2048-token context. The fused ESSH step uses `torch.compile`; baseline steps are not wrapped in `torch.compile`. Eager and CUDA-graph timing are reported separately, with graph capture applied to every method in the graph-mode comparison. Prefill processes the stated prompt at batch one.

The main 1.5B A100 fused-decode measurements use three independent sessions with 50 warm-up and 200 timed iterations, reporting the per-entry session median. The 370M family sweep uses three sessions with 20 warm-up and 100 timed iterations. The unfused A100 control uses one session with 20 warm-up and 100 timed iterations. The 370M sweep measures whole-model exported shapes with synthetic weights, while quality is evaluated from trained checkpoints. Each systems table identifies its device and execution settings.

Matched full-model measurements on A100, H200 and B300 use three sessions. Decode uses CUDA graphs with 50 warm-ups and 200 timed steps; prefill uses three warm-ups and five timed calls. Baselines follow the same per-device protocol. Compact measurements with one session are marked in their table captions. Ratios use unrounded measurements.

E.6 Baselines and parameter matching

The baselines address three comparison questions: full-capacity architecture quality, train-once compact families, and the quality obtainable from a dedicated training run at one deployment size. Parameter counts include embeddings and refer to the evaluated configuration. Published results are identified separately from locally trained or locally evaluated models. The 1.5B published quality rows come from the 100B-token study of Lahoti et al. (2026). Official release cards report 1.49B (State Space Models, 2026b) and 1.50B (State Space Models, 2026a) parameters for Mamba-3 SISO and MIMO, respectively; the remaining published rows retain the source's nominal ~ 1.5B size. The systems configurations are listed separately in Table 54. In the systems tables, Mamba-2/3 + FFN denote the official recurrent mixers alternating with residual feed-forward blocks. The systems Transformer++ uses the same 28-layer, width-2048 comparison setup. Our Transformer++ baseline uses causal full attention with RoPE, RMSNorm, and SwiGLU (Su et al., 2024; Zhang & Sennrich, 2019; Shazeer, 2020).

E.7 Post-hoc elastification protocol

The post-hoc study uses two pretrained parents: Mamba-3 + SWA trained for 7.4B tokens and the ESSH dense control trained for 3.7B tokens. We adapt the ranking-and-distillation procedure of Nemotron Elastic (Taghibakhshi et al., 2025a) to the fixed-depth comparison used here.

A 2M-token calibration subset supplies importance scores for feed-forward units in both parents. For Mamba-3 + SWA, it also scores recurrent heads and attention heads before constructing nested masks, following group-aware pruning principles (Taghibakhshi et al., 2025b). For the dense ESSH parent, spectral channels retain their original ordering and all attention heads are preserved. Depth and residual width remain fixed.

A frozen parent supervises the masked model for 0.5B recovery tokens using forward KL and label cross-entropy, both with coefficient one. Masks are sampled uniformly within each experiment. The dense-control recovery uses five masks: full capacity, approximately 75% and 50% of the parameters, and targets near 240.3M and 162.1M. The Mamba-3 comparisons use three masks: full capacity and the two targets near 185.3M and 274.3M. Recovery uses a 1% warm-up followed by constant learning rate 10^{-4}, with context length 2048. Compact comparisons target the T_4 and T_7 deployment sizes, with actual masked parameter counts reported for each export. Appendix N.5 provides a matched control that preserves the parent attention layers during recovery.

E.8 Seeds

Unless a table specifies another seed, the 370M language-model results use seed 197. The elastic-family comparison includes ESSH seeds 197 and 511, MatFormer seeds 197, 313, and 511, and

Table 8: Baseline roles and matching rules. Each method keeps its stated architecture and training budget. Provenance markers are defined in Appendix E.9.

Model	Used in	Matching rule
Transformer++, Gated DeltaNet, Mamba-2, Mamba-3 (SISO, MIMO), 1.5B, published†	Section 5.1, Appendix F	published source configurations and source-evaluation metrics; same corpus, tokenizer, context, and token budget where reported
Transformer++, own, 370M	Section 5.2, Appendices F, J	same depth, data, token budget, and schedule; parameter count within 2%
Gated DeltaNet + SWA, own, 370M	same	parameter count within 2%; its own depth, width, and SWA configuration
Mamba-3 + SWA⋆, own, 370M	same	dedicated full-capacity training; parameter count within 2%, with its own backbone configuration
MatFormer, own, 370M	Section 5.2, Appendix N	Transformer++ backbone with nested FFN widths and four sampled training capacities
MatMamba, own, 370M	same	Mamba-2 backbone with head-prefix nesting and four sampled capacities
Post-hoc elastic, Mamba-3 + SWA parent, own, 370M	Appendix N	post-hoc ranking and recovery from the pretrained reference hybrid, targeting the T_4 and T_7 deployment sizes
Post-hoc elastic, dense-control parent, own, 370M	same	same recovery procedure applied to the ESSH dense-control architecture
Dense control, own, 370M	Section 5.2, Appendices M, N	ESSH trained only at full capacity, then truncated and recovered for 2060 steps as in Table 43
Label-only prune-and-recover, own, 370M	Appendix N	fixed-shape recovery from the 3.7B dense control using label cross-entropy; no distillation or full-model label-loss term
Dense specialists◇, own, 370M	same	exact T_1, T_4, and T_7 export architectures trained independently for 7.4B tokens
Transformer, Mamba-3, Gated DeltaNet at Zoology size (Arora et al., 2024a), own	Appendix I	same generator, depth, residual width, training steps, and learning-rate grid; parameter counts reported separately
CNN, HyenaDNA, Mamba, Caduceus, published† (Grešová et al., 2023; Schiff et al., 2024)	Appendix K	published benchmark results at reported model sizes; bidirectional models identified separately
Four independent Transformer++ sizes, 370M study	Figure 3(b,c), Table 41	7.4B pretraining tokens per size; actual parameter counts near ESSH $T_1/T_4/T_7/T_{10}$
Mamba-2/3 + FFN and Transformer++, 1.53B systems	Figure 4, Appendix R	matched depth, residual width and parameter counts; configurations in Table 54

Table 9: Configurations used for the approximately 370M quality comparison. Attention positions are zero based. Pretraining KL denotes the full-tier distillation coefficient.

Model	Params (M)	Layers	Width	FFN	Attention blocks	Pretrain KL
ESSH	370.9	22	896	4608	12, 17, 21	0.5
Mamba-3 + SWA	377.1	24	1024	2304	10, 22	–
Gated DeltaNet + SWA	378.1	24	1024	2432	10, 22	–
Transformer++	365.6	22	896	4224	All 22	–
MatFormer	365.6	22	896	4224	All 22	0
MatMamba	369.9	48	1024	–	None	0

MatMamba seeds 197 and 313. The 3.7B ESSH control uses seeds 197, 313 and 511, and the dense controls use two seeds per pretraining budget. The initialization seeds are listed in Table 29.

The initialization study uses seeds 197 and 313 at 3.7B tokens. Seed-level values are reported directly; two-run ranges are descriptive and are not treated as confidence intervals.

E.9 Baselines, row markers, and cost accounting

Bold identifies the best reported metric within a common task, scale, or hardware comparison, including ties at the displayed precision. Size-specific recovery comparisons are ranked within tier, while seed-control tables keep the seed or tier fixed. Unmeasured entries are left blank.

Markers distinguish provenance: $\star$ identifies the locally trained Mamba-3 + SWA reference hybrid, † a published result, ‡ a public checkpoint evaluated with our harness, and $\diamond$ an independently trained specialist.

The primary quality frontier uses one final ESSH checkpoint for both full and compact tiers. Tables using raw checkpoints or other training stages identify them explicitly. Deployment tier, retained channel count, and actual parameter count are reported together for compact models.

F Complete Main Results

This section reports the complete public ESSH capacity family together with the matched controls and reference models used in the main comparisons.

F.1 The ten tiers at 370M

The ten tiers trace a smooth PPL–size frontier, while task accuracy and retrieval degrade more slowly across intermediate capacities (Table 10). With 26% fewer parameters, T_7 changes the seven-task mean from 0.428 to 0.420, while PPL increases from 14.71 to 15.91. At approximately half the full parameter count, T_4 retains a seven-task mean of 0.405 versus 0.428 and teacher-forced full-key accuracy of 0.370 versus 0.389.

PPL decreases monotonically as capacity increases. The seven-task mean and full-key accuracy are flatter over the intermediate tiers and need not vary monotonically; for example, T_9 reaches full-key accuracy 0.394, compared with 0.389 at full capacity. The metrics therefore have different capacity sensitivities, so the appropriate deployment tier depends on the target quality measure as well as model size.

Table 11 isolates the quality cost of sharing parameters across capacities within the ESSH architecture. Across all four matched shapes, the seven-task gap between the shared exports and independently trained models is at most 0.009, and the shared full-capacity model is slightly higher on this metric. Independent training nevertheless gives lower PPL at every shape, with the largest difference at T_1. Appendix N.7 compares the corresponding training cost of producing the four model sizes.

F.2 The 1.5B model

Throughout the 1.5B study, ESSH denotes the final model produced by the complete training procedure in Appendix D.8; every deployment tier uses that model.

The family extends the size–quality trade-off to the larger backbone and vocabulary. At T_7, 23.7% fewer parameters retain a seven-task mean of 53.01% compared with 55.03% at full capacity. The 0.828B T_4 retains 50.72% with 45.9% fewer parameters, while T_1 reaches 43.83% at 0.474B. These intermediate models provide substantial parameter savings before the larger accuracy reduction at the smallest tier.

The smallest export retains 31.0% of the full model's parameters, compared with 24.8% at 370M, because the larger vocabulary increases the fixed embedding cost. Tables 12 and 13 give the selected tiers, and Table 14 reports all ten. Published full-model comparisons remain in Table 1(a).

Table 13 separates the task-level responses to capacity reduction. Table 14 combines model quality with the measured deployment costs at the corresponding sizes.

Table 10: Complete 370M ESSH family and full-capacity local references. ESSH quality uses the final shared checkpoint after 7.4B-token pretraining and brief shared refinement; local full-model references use 7.4B training tokens. Retained fraction r and ΔNLL are relative to ESSH T_{10}. Full-key is teacher-forced full-key accuracy. Latency is a separate A100 batch-one shape benchmark on the exact ESSH export architectures with synthetic weights, a 2048-token context, and CUDA Graph execution.

Model / tier	Params (M)	r	PPL ↓	ΔNLL ↓	7-task ↑	Full key ↑	ms/token ↓
ESSH, T_1	92.1	0.248	23.61	0.473	0.376	0.241	**0.91**
ESSH, T_2	118.9	0.321	20.04	0.309	0.388	0.342	0.94
ESSH, T_3	154.7	0.417	18.45	0.227	0.402	0.355	1.01
ESSH, T_4	185.3	0.500	17.68	0.184	0.405	0.370	1.04
ESSH, T_5	214.5	0.578	16.98	0.144	0.413	0.366	1.10
ESSH, T_6	248.8	0.671	16.25	0.100	0.417	0.362	1.11
ESSH, T_7	274.3	0.740	15.91	0.078	0.420	0.366	1.13
ESSH, T_8	308.6	0.832	15.34	0.042	0.420	0.375	1.23
ESSH, T_9	337.9	0.911	15.08	0.025	0.424	**0.394**	1.24
ESSH, T_{10} (full)	370.9	1.000	14.71	0.000	0.428	0.389	1.31
Mamba-3 + SWA*	377.1	–	**14.10**	–	**0.445**	0.230	–
Gated DeltaNet + SWA	378.1	–	14.31	–	0.433	0.100	–
Transformer++	365.6	–	15.88	–	0.414	0.350	–
MatFormer, full	365.6	–	16.80	–	0.411	0.266	–
MatMamba, full	369.9	–	16.98	–	0.404	0.004	–

Table 11: Shared versus independently trained ESSH models at identical deployment shapes. The shared exports come from one 7.67B-token training run, while each independent model is trained separately for 7.4B tokens at the corresponding shape.

		PPL ↓		7-task ↑	
Tier	M params	Shared	Independent	Shared	Independent
T_1	92.1	23.61	**19.79**	0.376	**0.385**
T_4	185.3	17.68	**16.25**	0.405	**0.412**
T_7	274.3	15.91	**15.10**	0.420	**0.424**
T_{10}	370.9	14.71	**14.44**	**0.428**	0.425

Table 12: Final 1.5B ESSH at four deployment tiers. Accuracy and seven-task means are percentages. PPL uses the 1.5B validation stream. Published full-model references are in Table 1(a).

Tier	Params (B)	FW-Edu PPL ↓	LAMB PPL ↓	LAMB acc. ↑	7-task ↑
ESSH					
T_{10}	1.53	**10.11**	**12.96**	**46.8**	**55.0**
T_7	1.17	10.97	15.39	44.9	53.0
T_4	0.83	12.26	18.27	42.1	50.7
T_1	0.47	16.49	36.87	34.0	43.8

Table 13: Seven-task accuracy (%) for the final 1.5B ESSH model. HellaSwag and ARC-Challenge use normalized accuracy; other columns use accuracy. Published full-model results are listed in Table 1(a).

Model / tier	Params (B)	LAMB	HellaS	PIQA	ARC-e	ARC-c	Wino	OBQA
ESSH, T_{10}	1.53	**46.8**	**60.6**	**72.9**	**73.7**	**40.7**	58.3	**32.2**
ESSH, T_7	1.17	44.9	56.6	71.9	70.6	38.1	58.4	30.6
ESSH, T_4	0.83	42.1	51.9	69.4	67.5	35.1	**58.6**	30.4
ESSH, T_1	0.47	34.0	39.6	66.5	59.8	29.5	52.7	24.6

Table 14: 1.5B deployment family. Quality uses one ESSH checkpoint across all ten tiers; runtime uses B300 shape benchmarks. Prefill is batch one at 2K; decode is batch one after 2K. Decode uses three sessions at every tier; prefill uses three sessions at T_{10} and one session at compact tiers. Persistent memory is state+KV, excluding weights. Selected tiers are summarized in Table 1(b).

tier	params (B)	PPL ↓	7-task ↑	Full key ↑	prefill ktok/s ↑	decode ms/token ↓	memory MB ↓
T_1	0.474	16.49	0.438	0.370	75.0	1.13	**51.15**
T_2	0.575	14.08	0.462	**0.384**	79.8	**1.12**	51.46
T_3	0.716	12.87	0.486	0.367	77.9	1.13	51.87
T_4	0.828	12.26	0.507	0.379	**81.4**	1.20	52.17
T_5	0.931	11.78	0.515	0.371	74.6	1.18	52.38
T_6	1.076	11.23	0.526	0.323	76.1	1.28	52.69
T_7	1.168	10.97	0.530	0.339	74.0	1.34	52.89
T_8	1.291	10.63	0.538	0.321	76.9	1.33	53.20
T_9	1.416	10.35	0.547	0.334	76.1	1.36	53.40
T_{10}	1.531	**10.11**	**0.550**	0.330	71.0	1.37	53.61

G Downstream Language Understanding

Per-task results complement the aggregate seven-task mean by showing how individual benchmarks respond to capacity reduction.

G.1 370M per-task results

Table 15 compares individual task scores. Full ESSH exceeds Transformer++ on six of the seven tasks. Mamba-3 + SWA has the highest mean, while ESSH is higher on ARC-Challenge, 0.286 versus 0.270. Capacity reduction affects the tasks differently. At 214.5M parameters, T_5 reaches a seven-task mean of 0.413, compared with 0.414 for the 365.6M Transformer++, despite using approximately 41% fewer parameters. Their nearly identical means arise from different task profiles: T_5 is higher on ARC-Easy and ARC-Challenge and equal on PIQA, while Transformer++ is higher on LAMBADA, HellaSwag, and OpenBookQA. WinoGrande is close to chance for both models.

Within the ESSH family, LAMBADA and HellaSwag show the clearest upward trend with capacity. PIQA is comparatively flat over the upper tiers: T_7 scores 0.665, only 0.005 below the full-capacity score of 0.670. ARC-Easy and ARC-Challenge vary less regularly across the intermediate tiers, while WinoGrande remains near chance and should not be interpreted through relative-retention ratios. Reducing from T_{10} to T_1 changes LAMBADA and HellaSwag accuracy by 0.098 and 0.096, respectively, while each of the other five tasks changes by at most 0.070.

Table 15: Zero-shot accuracy for the 370M comparison. ESSH rows use the final shared checkpoint and correspond to the ten deployment tiers; local reference models are trained independently. HellaSwag and ARC-Challenge use normalized accuracy. Scores are reported to three decimals.

		accuracy ↑							
Model	params (M)	LAMB acc	HellaS	PIQA	ARC-e	ARC-c	Wino	OBQA	mean
ESSH, T_1	92.1	0.230	0.308	0.626	0.508	0.242	0.535	0.180	0.376
ESSH, T_2	118.9	0.253	0.322	0.644	0.540	0.259	0.515	0.186	0.388
ESSH, T_3	154.7	0.269	0.344	0.656	0.561	0.275	0.516	0.190	0.402
ESSH, T_4	185.3	0.275	0.355	0.660	0.555	0.275	0.512	0.200	0.405
ESSH, T_5	214.5	0.293	0.365	0.653	0.574	0.282	0.513	0.210	0.413
ESSH, T_6	248.8	0.305	0.379	0.660	0.574	0.286	0.509	0.206	0.417
ESSH, T_7	274.3	0.309	0.387	0.665	0.577	0.284	0.514	0.204	0.420
ESSH, T_8	308.6	0.316	0.391	0.670	0.574	0.285	0.503	0.204	0.420
ESSH, T_9	337.9	0.329	0.399	0.671	0.574	0.277	0.519	0.202	0.424
ESSH, T_{10}	370.9	0.328	0.404	0.670	0.578	0.286	0.517	0.216	0.428
Mamba-3 + SWA* (reference hybrid)	377.1	0.345	0.421	0.675	0.619	0.270	0.534	0.250	0.445
Transformer++	365.6	0.314	0.375	0.653	0.568	0.263	0.501	0.222	0.414
Gated DeltaNet + SWA	378.1	0.333	0.415	0.676	0.587	0.273	0.519	0.228	0.433

Task accuracy across tiers 370M, zero-shot accuracy ↑ fill = change from T_{10}: −0.1 +0.1

	LAMB	HellaS	PIQA	ARC-e	ARC-c	Wino	OBQA	mean
Ours T_{10}	.328	.404	.670	.578	.286	.517	.216	.428
Ours T_7	.309	.387	.665	.577	.284	.514	.204	.420
Ours T_4	.275	.355	.660	.555	.275	.512	.200	.405
Ours T_1	.230	.308	.626	.508	.242	.535	.180	.376
Mamba-3 + SWA	.345	.421	.675	.619	.270	.534	.250	.445
Transformer++	.314	.375	.653	.568	.263	.501	.222	.414
GDN + SWA	.333	.415	.676	.587	.273	.519	.228	.433

Figure 6: Task-level zero-shot accuracy across ESSH capacities (T_1, T_4, T_7, T_{10}) and three independently trained full-size references. Cell values report absolute accuracy, and Mean is the unweighted average of the seven tasks. Shading for ESSH tiers represents the absolute score change relative to T_{10} within each task; reference rows are unshaded. Table 15 gives parameter counts and complete numerical results.

G.2 1.5B PER-TASK RESULTS

The six additional tasks give a complementary view of capacity selection (Table 16). At T_7, the mean is 0.547 compared with 0.549 at full capacity, and SciQ remains at 0.904 versus 0.911. The intermediate export therefore preserves a similar aggregate score with 23.7% fewer parameters.

Table 16: Additional zero-shot task accuracies (fractions, ↑) for the final 1.5B ESSH model. Mean averages the six task columns. The seven reference tasks are in Table 13.

Model / tier	Params (B)	BoolQ	SIQA	SciQ	COPA	RACE	LogiQA	Mean
ESSH, T_{10}	1.531	0.613	0.419	**0.911**	0.790	**0.344**	0.214	**0.549**
ESSH, T_7	1.168	0.594	**0.422**	0.904	**0.810**	**0.344**	0.209	0.547
ESSH, T_4	0.828	0.596	0.400	0.890	0.740	**0.344**	0.198	0.528
ESSH, T_1	0.474	**0.617**	0.395	0.870	0.710	0.297	**0.220**	0.518

H RETRIEVAL AND LENGTH

Retrieval is evaluated separately from language-model perplexity. The needle probe, open-book tasks, and synthetic suite test different behaviors and use different scoring rules.

H.1 NEEDLE RECALL BY TIER

The 2048-token NIAH test measures preference for a planted key over a counterfactual key, first-token prediction, and full-key accuracy (Table 17). Full-key accuracy is reported in Table 1(b) as a percentage. At 1.5B, the selected ESSH tiers score 0.330–0.379. The 0.828B T_4 reaches 0.379 and the 0.474B T_1 reaches 0.370, preserving useful key retrieval after substantial parameter reduction.

At 370M, full-key accuracy is 0.370 for the half-size T_4 and 0.389 at full capacity, while the smallest T_1 reaches 0.241. Moderate reduction retains much of this capability, but more aggressive reduction incurs a larger retrieval loss even though the attention layout is fixed. Model scales use different tokenizers and are compared within scale.

The 370M T_1 ranks the planted key above its counterfactual in 0.845 of trials but predicts the full key correctly in 0.241. Relative preference and exact token prediction therefore respond differently to capacity reduction; the full-key metric directly evaluates the latter.

H.2 BY DEPTH AND BY CONTEXT LENGTH

Table 18 varies the key depth at 2048 tokens and the context length at the middle depth for the final 1.5B ESSH model. The depth dependence persists across capacities: accuracy at the most recent placement is 0.633–0.754, compared with 0.004–0.051 at the earliest placement.

Table 17: NIAH retrieval at context 2048, using 256 key pairs at five depths. Full-key accuracy uses teacher forcing. First-token accuracy and candidate margin are separate fields. The final column is the half-width of the 95% item-cluster bootstrap interval for full-key accuracy (2000 resamples of key pairs, keeping their five depths together). The 1.5B block uses the final ESSH model. Full key is also reported in Table 1(b), in percent.

Model	tier	params (M)	margin win ↑	first token ↑	full key ↑	95% half-width
ESSH	T_{10}	1531.1	0.826	0.332	0.330	0.039
ESSH	T_7	1168.3	0.831	0.341	0.339	0.039
ESSH	T_4	827.6	**0.857**	**0.380**	**0.379**	0.040
ESSH	T_1	474.5	0.845	0.372	0.370	0.038
ESSH 370M	T_{10}	370.9	0.826	**0.389**	**0.389**	0.036
ESSH 370M	T_7	274.3	0.824	0.366	0.366	0.034
ESSH 370M	T_4	185.3	0.828	0.370	0.370	0.035
ESSH 370M	T_1	92.1	**0.845**	0.242	0.241	0.025
Mamba-3 + SWA*	–	377.1	0.738	0.241	0.230	0.023
Transformer++	–	365.6	0.799	0.350	0.350	0.030
Gated DeltaNet + SWA	–	378.1	0.771	0.100	0.100	0.018

At the middle depth, shorter contexts give higher full-key accuracy. The following subsection evaluates next-token PPL over longer windows as a separate capability.

Table 18: Teacher-forced full-key accuracy for the final 1.5B ESSH model. Left: five needle depths in a 2048-token sequence. Right: sequence lengths with the needle at depth 0.5. Each depth uses 256 trials; 2048 is the same depth-0.5 measurement in both panels.

	Depth at 2048 tokens					Sequence length (tokens)		
Tier	0.1	0.3	0.5	0.7	0.9	512	1024	2048
ESSH								
T_1	0.004	0.242	**0.355**	0.496	**0.754**	**0.742**	**0.582**	**0.355**
T_4	**0.051**	**0.270**	0.352	**0.500**	0.723	0.711	**0.582**	0.352
T_7	0.027	0.246	0.324	0.457	0.641	0.668	0.547	0.324
T_{10}	0.035	0.227	0.309	0.445	0.633	0.664	0.512	0.309

H.3 Perplexity beyond the training length

The length sweep tests whether capacity reduction introduces additional degradation when evaluation extends beyond the 2048-token training context. Each length from 2K to 16K is evaluated on its own set of 64 non-overlapping windows.

The ordering of the four ESSH tiers is unchanged at every tested length (Table 19). PPL also remains in a similar range across the sweep. The 2K and 16K endpoint values are 14.91 and 15.05 for T_{10}, 16.03 and 16.23 for T_7, 17.76 and 17.95 for T_4, and 23.37 and 23.70 for T_1. Because each length uses a different set of evaluation windows, these endpoint differences include variation in the evaluated text as well as any effect of sequence length.

The compact-to-full gaps remain similar across the tested lengths. For example, the T_4–T_{10} PPL gap is 2.85 at 2K and 2.90 at 16K, while the T_7–T_{10} gap changes from 1.12 to 1.18. Capacity reduction therefore does not introduce a marked additional PPL gap over this length range.

The selective recurrence applies the same state update at every position, and the shared attention window remains fixed. The recurrent update and the 2048-token attention window keep the same local operations as context length increases. The Mamba-3 + SWA reference shows a similarly narrow range of PPL values across the same tested lengths. The 1.5B ESSH family likewise preserves its tier ordering from 2K to 16K. Full-model PPL changes from 10.28 to 10.99 and T_4 from 12.45 to 13.40; their gap changes from 2.17 to 2.41. The largest tier remains the strongest language model, while the intermediate export retains a similar additional loss across this length range.

Longer-window PPL measures next-token prediction over longer inputs, separately from the key-retrieval measurements above.

Table 19: Perplexity beyond the 2048-token training context. Each length is evaluated on its own set of 64 non-overlapping windows at each scale. Both ESSH scales use their final models.

		validation PPL ↓ by window length			
Model	params (M)	2048	4096	8192	16384
ESSH, T_{10}	370.9	14.91	14.97	14.76	15.05
ESSH, T_7	274.3	16.03	16.13	15.92	16.23
ESSH, T_4	185.3	17.76	17.83	17.62	17.95
ESSH, T_1	92.1	23.37	23.42	23.22	23.70
Mamba-3 + SWA 370M* (reference hybrid)	377.1	14.34	14.41	14.19	14.46
1.5B ESSH					
T_{10}	1531.1	10.28	9.92	10.89	10.99
T_7	1168.3	11.16	10.76	11.78	11.91
T_4	827.6	12.45	12.03	13.22	13.40
T_1	474.5	16.57	16.20	18.22	18.81

H.4 Recall-intensive open-book tasks

We use the six-task recall-intensive suite described by JRT (Arora et al., 2024c), extending the extraction and question-answering evaluations in Based (Arora et al., 2024b). All tasks use contains-match accuracy with a 2048-token context. The underlying datasets are SWDE, FDA, SQuAD, TriviaQA, Natural Questions, and DROP (Hao et al., 2011; Arora et al., 2025; Rajpurkar et al., 2016; Joshi et al., 2017; Kwiatkowski et al., 2019; Dua et al., 2019).

At 370M, ESSH has the highest scores on SWDE, FDA and SQuAD and the highest six-task mean, 0.253 (Table 20; Figure 7). Its half-size T_4 retains a mean of 0.225, above the full Mamba-3 + SWA reference at 0.217 and Gated DeltaNet + SWA at 0.221. Thus the compact export preserves a useful advantage on these supplied-context tasks despite its reduced parameter count.

At 1.5B, the mean rises from 0.302 at T_1 to 0.394 at full capacity. SWDE, TriviaQA and NQ increase with capacity, while SQuAD and DROP reach their highest reported scores at T_4 (0.453 and 0.259). The intermediate tiers therefore preserve substantial extraction and question-answering performance without requiring every task to improve monotonically with model size.

Table 20: Contains-match accuracy on six open-book tasks at context 2048. Locally trained 370M rows share data and pretraining tokens. The 1.5B group uses ESSH; model scales are ranked separately.

		contains-match accuracy ↑						
Model	Params (M)	SWDE	FDA	SQuAD	TriviaQA	NQ	DROP	mean
1.5B ESSH								
ESSH, T_{10}	1531.1	**0.491**	**0.358**	0.446	**0.629**	**0.189**	0.251	**0.394**
ESSH, T_7	1168.3	0.452	0.282	0.446	0.618	0.173	0.251	0.371
ESSH, T_4	827.6	0.411	0.297	**0.453**	0.578	0.151	**0.259**	0.358
ESSH, T_1	474.5	0.365	0.216	0.394	0.515	0.111	0.208	0.302
370M models								
ESSH, T_{10}	370.9	**0.290**	**0.144**	**0.334**	0.477	0.097	0.178	**0.253**
ESSH, T_4	185.3	0.280	0.074	0.311	0.428	0.083	0.175	0.225
Mamba-3 + SWA*	377.1	0.155	0.077	0.297	**0.482**	0.098	**0.196**	0.217
Gated DeltaNet + SWA	378.1	0.166	0.083	0.308	0.477	**0.104**	0.189	0.221
Transformer++	365.6	0.279	0.088	0.303	0.459	0.091	0.159	0.230

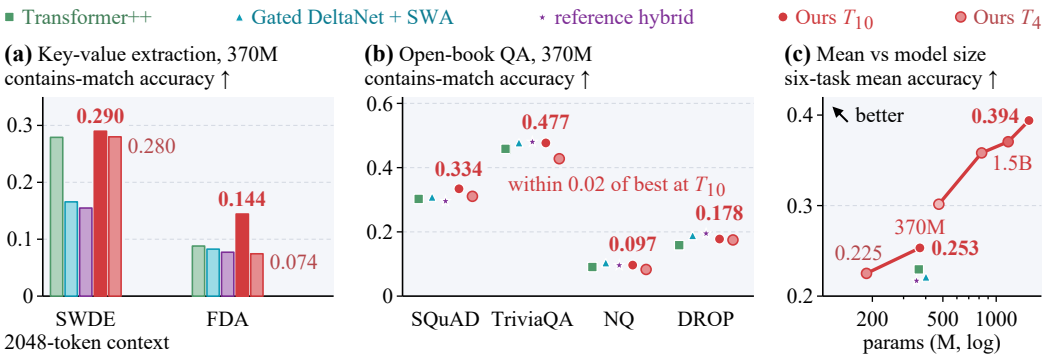

Figure 7: Retrieval and question answering from a supplied 2048-token context. (a) SWDE and FDA test extraction of values. (b) four open-book QA tasks test answers recovered from the context. Scores use contains-match accuracy, with higher values better. Generation is greedy, stops at a newline and is capped at 48 tokens; inputs are left-truncated to at most 2000 tokens within the 2048-token evaluation budget. Panel (c) plots the mean of these six tasks against parameter count for the 370M and 1.5B ESSH families, so quality should be compared at similar sizes; the nearly coincident GDN + SWA and Mamba-3 + SWA points are offset slightly along the x-axis. The reference hybrid denotes Mamba-3 + SWA. Table 20 gives the complete 1.5B and 370M results.

I Synthetic Recall Suite

The synthetic suite isolates recall-intensive behaviors in small task-specific models and tests whether they remain available under the same spectral-channel and feed-forward truncation used by the language models. Each backbone applies the public capacity map to its own dimensions and uses the stated channel value width P. In the two-layer synthetic hybrid, the final block uses attention with a window covering the full input, shared across tiers.

Multi-query associative recall. MQAR uses two-layer, width-128 backbones with feed-forward width 512 (Arora et al., 2024a); ESSH uses channel value width $P = 24$. For ESSH, the learning rate for each sequence-length and pair-count setting is selected before the matched rerun and then held fixed while evaluating T_1, T_4, and T_{10} from the same 20k-step checkpoint. Baseline rows report the best result from their learning-rate sweeps.

Across three seeds, the hybrid retains strong recall at 64 pairs: mean scaled accuracy remains between 0.971 and 1.000 across the three lengths and all reported tiers (Table 22). At $L = 1024$, reducing the model from T_{10} to T_1 changes the mean from 0.981 to 0.971. The shared attention path remains present at every tier, so substantial channel and FFN reduction preserves this recall behavior in the tested hybrid. Mixer-only models remain near zero in these settings despite having more full-capacity parameters. This supports the complete hybrid on the tested recall task, while the Transformer results show that attention alone is not sufficient under the evaluated configuration.

The repeated runs also distinguish capacity effects from training variability. With 32 pairs at $L = 512$, the three ESSH T_{10} scores span 0.019–0.994, and the compact tiers track the full model within each run. Mamba-3 likewise varies substantially across seeds in several 32-pair settings. Thus the stable 64-pair results support preservation of the hybrid's recall under truncation, while the 32-pair averages show that changing capacity within one trained model has a different effect from changing its training seed. Table 21 reports the seed-0 settings for the displayed models; Table 22 reports the matched repeats.

I.1 Palindrome Completion

Palindrome completion extends associative retrieval from individual key–value pairs to reproducing an entire input sequence in reverse order. At $L = 1024$, the full-capacity hybrid scores 1.000 and the T_4 export scores 0.997. The full-capacity mixer-only model has more parameters, 0.996M versus

Table 21: MQAR scaled accuracy after 20k training steps, seed 0. Three-seed results for 32 and 64 pairs are in Table 22. For each length/pair setting, ESSH uses one matched run with a preselected learning rate and evaluates T_1, T_4, and T_{10} from the same checkpoint. Baseline rows report their best learning-rate sweep result. Columns vary sequence length and key–value pair count; parameter counts use the task-specific vocabulary. Bold marks the highest displayed accuracy within each setting.

Sequence length $L = 256$, scaled accuracy ↑						
Model	Capacity	Params (M)	8 pairs	16 pairs	32 pairs	64 pairs
ESSH	T_{10}	1.81	0.999	0.998	**0.999**	**1.000**
ESSH	T_4	1.43	0.998	0.998	0.998	**1.000**
ESSH	T_1	1.24	0.998	0.999	**0.999**	**1.000**
ESSH (mixer only)	T_{10}	2.04	0.124	0.063	0.024	0.004
ESSH (mixer only)	T_4	1.53	0.128	0.062	0.023	0.003
ESSH (mixer only)	T_1	1.25	0.077	0.003	0.011	0.003
Transformer	–	1.57	**1.000**	**1.000**	0.022	0.008
Mamba-3	–	1.72	**1.000**	**1.000**	0.923	0.010

Sequence length $L = 512$, scaled accuracy ↑						
Model	Capacity	Params (M)	8 pairs	16 pairs	32 pairs	64 pairs
ESSH	T_{10}	1.81	0.998	0.874	**0.990**	**0.992**
ESSH	T_4	1.43	0.994	0.631	0.986	**0.992**
ESSH	T_1	1.24	0.997	0.650	0.978	0.989
ESSH (mixer only)	T_{10}	2.04	0.125	0.059	0.003	0.000
ESSH (mixer only)	T_4	1.53	0.106	0.055	0.003	0.000
ESSH (mixer only)	T_1	1.25	0.053	0.017	0.002	0.000
Transformer	–	1.57	**1.000**	**1.000**	0.025	0.006
Mamba-3	–	1.72	0.991	**1.000**	0.825	0.008

Sequence length $L = 1024$, scaled accuracy ↑						
Model	Capacity	Params (M)	8 pairs	16 pairs	32 pairs	64 pairs
ESSH	T_{10}	1.81	0.995	0.992	**0.953**	0.978
ESSH	T_4	1.43	0.997	0.990	0.923	**0.979**
ESSH	T_1	1.24	0.993	0.983	0.906	0.977
ESSH (mixer only)	T_{10}	2.04	0.051	0.047	0.002	0.000
ESSH (mixer only)	T_4	1.53	0.015	0.014	0.002	0.000
ESSH (mixer only)	T_1	1.25	0.016	0.001	0.000	0.000
Transformer	–	1.57	**1.000**	**1.000**	0.027	0.013
Mamba-3	–	1.72	**1.000**	**1.000**	0.024	0.119

Table 22: MQAR across training seeds 0, 1, and 2: mean [minimum, maximum] scaled accuracy after 20k steps. The learning rate is fixed across seeds within each model/length/pair setting. ESSH tiers share each seed's checkpoint. Bold marks the highest displayed mean within each length/pair setting.

Model / tier	$L = 256$	$L = 512$	$L = 1024$
32 key–value pairs			
ESSH T_{10}	**0.937** [0.818, 0.999]	**0.668** [0.019, 0.994]	**0.917** [0.811, 0.988]
ESSH T_4	0.846 [0.549, 0.998]	0.664 [0.014, 0.993]	0.854 [0.659, 0.980]
ESSH T_1	0.792 [0.632, 0.999]	0.661 [0.012, 0.991]	0.867 [0.728, 0.966]
Mixer only T_{10}	0.024 [0.022, 0.025]	0.005 [0.002, 0.009]	0.005 [0.002, 0.010]
Mixer only T_4	0.022 [0.017, 0.024]	0.002 [0.002, 0.003]	0.003 [0.002, 0.004]
Mixer only T_1	0.010 [0.009, 0.011]	0.002 [0.002, 0.002]	0.002 [0.000, 0.004]
Transformer	0.021 [0.021, 0.022]	0.025 [0.025, 0.025]	0.026 [0.025, 0.027]
Mamba-3	0.323 [0.022, 0.923]	0.290 [0.022, 0.825]	0.349 [0.024, 0.997]
64 key–value pairs			
ESSH T_{10}	**1.000** [1.000, 1.000]	**0.991** [0.989, 0.992]	**0.981** [0.978, 0.989]
ESSH T_4	**1.000** [1.000, 1.000]	0.990 [0.987, 0.992]	**0.981** [0.974, 0.989]
ESSH T_1	**1.000** [1.000, 1.000]	0.985 [0.974, 0.992]	0.971 [0.967, 0.977]
Mixer only T_{10}	0.008 [0.004, 0.011]	0.002 [0.000, 0.005]	0.000 [0.000, 0.001]
Mixer only T_4	0.006 [0.003, 0.008]	0.002 [0.000, 0.004]	0.000 [0.000, 0.001]
Mixer only T_1	0.004 [0.003, 0.005]	0.001 [0.000, 0.002]	0.000 [0.000, 0.001]
Transformer	0.008 [0.007, 0.008]	0.007 [0.006, 0.008]	0.013 [0.013, 0.013]
Mamba-3	0.009 [0.009, 0.010]	0.009 [0.008, 0.010]	0.364 [0.004, 0.969]

0.765M for the hybrid, but scores only 0.120 (Table 23). Its accuracy also falls sharply as sequence length increases, whereas the hybrid remains near perfect at T_{10} and T_4.

For each sequence-length setting, the attention window covers the full input and remains fixed across tiers, so the hybrid retains a direct attention path from each copying position to the source sequence. The contrast with the mixer-only model is consistent with this path being important for sequence copying. Because attention is shared across ESSH tiers, channel and FFN truncation reduce recurrent and tokenwise capacity without removing this direct source-access path. Even at T_1, which retains only 0.198M parameters, palindrome accuracy remains 0.974 at $L = 1024$.

Table 23: Palindrome token accuracy after 20k training steps, seed 0. Tier rows use the retained standalone model shape. Depth, residual width, and training steps are shared across the compared task-specific models; parameter counts differ.

Model	Capacity	Params (M)	$L = 256$	$L = 512$	$L = 1024$
Hybrid	T_{10}	0.765	1.000	1.000	1.000
Hybrid	T_4	0.389	1.000	1.000	0.997
Hybrid	T_1	0.198	0.999	0.996	0.974
Mixer only	T_{10}	0.996	0.892	0.428	0.120
Mixer only	T_4	0.491	0.088	0.111	0.040
Mixer only	T_1	0.207	0.147	0.105	0.044

J Cross-Domain Language Modeling

Table 24 evaluates transfer to six text domains using perplexity within each tokenizer group. All 370M models share the same pretraining corpus and budget; ESSH also receives brief shared refinement. The 1.5B block uses the final ESSH model.

ESSH has lower PPL than Transformer++ on all six 370M domain slices and the lowest PPL on arXiv and Proof-Pile (Table 24). The 274.3M T_7 also remains below the 377.1M Mamba-3 + SWA reference on arXiv, GitHub and Proof-Pile. Gated DeltaNet + SWA leads on PG-19, GitHub and WikiText, while Mamba-3 + SWA leads on CC-News, so the domain-level results complement the main-validation ranking.

Capacity sensitivity varies by domain. At 1.5B, reducing to T_1 raises GitHub PPL from 43.65 to 97.24, while CC-News changes from 13.89 to 22.79. The intermediate T_7 stays closer to full capacity at 50.63 and 14.93, respectively. These differences make the evaluated intermediate tiers useful for selecting a size under domain-specific quality requirements.

Table 24: Cross-domain perplexity (↓), compared within each tokenizer group. The 1.5B block uses ESSH; the 370M group uses the same training stages as the main comparison. Each group scores fixed non-overlapping 2048-target-token windows, up to 400 per domain; WikiText supplies 139 windows for GPT-2 and 141 for Llama-3.1. Bold compares models within a group.

Model	Params (M)	PG-19	arXiv	GitHub	CC-News	Proof-Pile	WikiText
1.5B ESSH; Llama-3.1 tokenizer							
ESSH, T_{10}	1531.1	**27.59**	**28.35**	**43.65**	**13.89**	**36.00**	**13.57**
ESSH, T_7	1168.3	30.11	31.16	50.63	14.93	39.21	14.88
ESSH, T_4	827.6	33.30	34.36	58.86	16.70	43.92	16.71
ESSH, T_1	474.5	48.40	48.61	97.24	22.79	63.94	23.35
370M models; GPT-2 tokenizer							
ESSH, T_{10}	370.9	31.34	**23.75**	14.28	20.62	**28.91**	22.01
ESSH, T_7	274.3	34.26	25.16	15.52	22.20	30.33	23.90
ESSH, T_4	185.3	37.22	27.25	16.73	24.62	33.04	26.72
ESSH, T_1	92.1	51.30	37.20	24.74	33.06	45.79	36.83
Mamba-3 + SWA	377.1	30.57	27.87	16.20	**19.89**	33.38	21.46
Gated DeltaNet + SWA	378.1	**30.12**	26.31	**14.04**	20.12	32.61	**20.93**
Transformer++	365.6	34.50	29.42	16.44	22.07	34.97	23.99

K DNA Sequence Modeling

The DNA study tests whether the same train-once, export-many capacity mechanism transfers beyond language modeling. We pretrain a four-layer, 0.534M-parameter ESSH backbone on GRCh38 for 3B nucleotide tokens, holding out the chr21 primary contig (Schneider et al., 2017). The DNA backbone contains spectral blocks without attention and uses channel value width $P = 8$, which determines its tier-specific parameter and recurrent-state counts.

Published DNA models at similar parameter scales are reported as external references in the downstream experiments. Their training recipes and directionality differ from ours, so controlled comparisons use the same ESSH backbone across capacities and pretraining conditions.

K.1 Held-out next-nucleotide loss

Chromosome 21 evaluates the pretrained exports before task-specific adaptation. Across the four measured tiers, next-nucleotide loss ranges from 1.775 bits per base at T_1 to 1.662 at full capacity (Table 25), but the loss–capacity curve is not uniform.

The middle two tiers are nearly identical on this metric: T_4 reaches 1.732 bits per base and T_7 reaches 1.729, a difference of only 0.003, although T_4 uses 34% fewer parameters. By contrast, increasing from T_7 to T_{10} reduces loss by 0.067 bits per base. The T_4 export therefore reaches nearly the same held-out loss as T_7 with 34% fewer parameters.

The next subsection evaluates the same pretrained exports after task-specific adaptation to genomic classification benchmarks.

Table 25: Held-out chromosome-21 next-nucleotide loss. Every tier is evaluated on the same 4000 non-overlapping windows of 1024 bases. Parameter counts exclude task-specific classification heads.

	T_1	T_4	T_7	T_{10}
Params (M)	0.135	0.240	0.362	0.534
Bits per base ↓	1.775	1.732	1.729	1.662

K.2 GENOMIC BENCHMARKS ACROSS CAPACITIES

Each export is adapted for ten epochs on eight of the nine Genomic Benchmarks tasks, with test accuracy taken at the best validation epoch and averaged over seeds 0, 197, and 511 (Grešová et al., 2023). All four exports are evaluated under this protocol.

The clearest control is the full architecture trained without pretraining. The pretrained full model improves on it in all eight tasks, and even the 0.135M T_1 exceeds this 0.534M control in every task. Shared pretraining therefore supplies useful sequence representations that survive substantial capacity reduction. This improvement remains available when roughly three quarters of the full backbone's parameters are removed.

Table 26 shows task-specific responses to capacity reduction. T_4 and T_7 exceed the full model on mouse enhancers and regulatory sequences, whereas non-TATA promoter accuracy increases from 0.887 at T_1 to 0.942 at T_{10}. Capacity selection therefore depends on the downstream task: some tasks retain their strongest measured score in a compact export, while the promoter task benefits consistently from additional capacity. These outcomes complement the held-out next-nucleotide loss, which favors the full model overall. Among the published references, ESSH has the highest coding-versus-intergenomic score (0.923), and every tier exceeds the reported reference scores on human regulatory sequences. The 0.135M T_1 exceeds the 0.264M CNN on all eight tasks. Caduceus leads the other six tasks using a bidirectional, reverse-complement-aware architecture. Published values are taken from the comparison reported by Schiff et al. (2024), including its HyenaDNA and Mamba runs; the CNN originates from Grešová et al. (2023).

Table 26: Genomic Benchmarks test accuracy (↑). ESSH rows average three task-adaptation seeds (0, 197, 511); each export is adapted separately. No pretraining uses the full architecture. †: results reported by Schiff et al. (2024), with the CNN from Grešová et al. (2023); bidir.: bidirectional. Published recipes differ from ours. Bold marks the best measured score within each block. OCR denotes open chromatin regions.

Model / tier	Params (M)	Mouse enh.	Coding	Human/ worm	Cohn	Ensembl enh.	Regul.	OCR	Non-TATA
T_1	0.135	0.759	0.905	**0.962**	**0.729**	0.834	0.879	0.765	0.887
T_4	0.240	0.774	0.906	0.960	0.721	0.836	**0.885**	0.761	0.913
T_7	0.362	**0.775**	0.906	0.960	0.725	0.846	0.884	0.766	0.925
T_{10}	0.534	0.760	**0.923**	0.961	0.726	**0.869**	0.879	**0.773**	**0.942**
No pretraining	0.534	0.716	0.871	0.949	0.682	0.773	0.745	0.683	0.808
CNN†	0.264	0.715	0.892	0.942	0.702	0.744	0.872	0.698	0.861
HyenaDNA†	0.436	0.780	0.904	0.964	0.729	0.849	0.869	0.783	0.944
Mamba†	0.468	0.743	0.904	0.967	0.732	0.862	0.814	0.815	0.933
Caduceus-Ph† (bidir.)	0.470	0.754	**0.915**	**0.973**	**0.747**	0.893	0.872	**0.828**	**0.946**
Caduceus-PS† (bidir.)	0.470	**0.793**	0.910	0.968	0.745	**0.900**	**0.873**	0.818	0.945

L SELECTIVE-DYNAMICS ABLATIONS

This section examines the selective recurrence, its initialization, and the allocation of recurrent state under the 370M training recipe. Clock and gate controls test input-dependent state dynamics; the remaining experiments vary the initial temporal representation, mode count, or channel partition. Capacity-map and distillation controls are reported separately in Appendix M.

L.1 CLOCK, GATES, AND INITIALIZATION

The control is the 370M ESSH recipe trained for 3.7B tokens with seed 197. The input-independent-clock arm fixes the clock projection weights at zero while retaining trainable clock biases. The neutral-gate arm fixes both write/read gates at one. State dimensions, the capacity map, and the attention layout are unchanged. Full-capacity PPL uses the end-of-pretraining model for comparison with full-capacity-only training; reduced-capacity PPL and retrieval use the final models. Comparisons between arms use the same stage within each column.

At full capacity, fixing the clock increases PPL by 0.34 and neutralizing the gates increases it by 0.63, compared with the 0.09 range across three control seeds (Table 29). Both interventions also

increase PPL at the reduced capacities. The corresponding T_4 first-token accuracies fall from 0.259 to 0.139 and 0.102. Thus language modeling and retrieval both favor the input-dependent controls in these runs, with retrieval reported as a single-seed measurement.

Table 27: Selective-dynamics controls at 370M after 3.7B pretraining tokens. Compact PPL and needle first-token accuracy use the final checkpoints; full-capacity PPL uses the corresponding raw checkpoint. Increasing m also increases parameter count and recurrent-state size.

	validation PPL ↓				needle first-token acc. ↑		7-task ↑	params
Arm	T_1	T_4	T_7	T_{10}	T_4	T_{10}	T_{10}	T_{10} (M)
ESSH (control)	25.32	19.15	17.38	16.46	0.259	0.336	0.408	370.9
ESSH (no selective clock)	25.69	19.47	17.70	16.80	0.139	0.266	0.410	370.9
ESSH (neutral gates)	26.04	19.79	17.99	17.09	0.102	0.140	0.406	370.9
ESSH (non-oscillatory init., no fit)	26.18	19.92	18.01	17.00	0.140	0.180	0.405	370.9
ESSH (random damped modes)	26.00	19.64	17.68	16.69	0.234	0.280	0.406	370.9
ESSH (shuffled slots)	25.66	19.31	17.51	16.51	0.115	0.202	0.414	370.9
ESSH ($m = 16$)	26.17	19.39	17.51	16.61	0.263	0.347	0.409	384.0

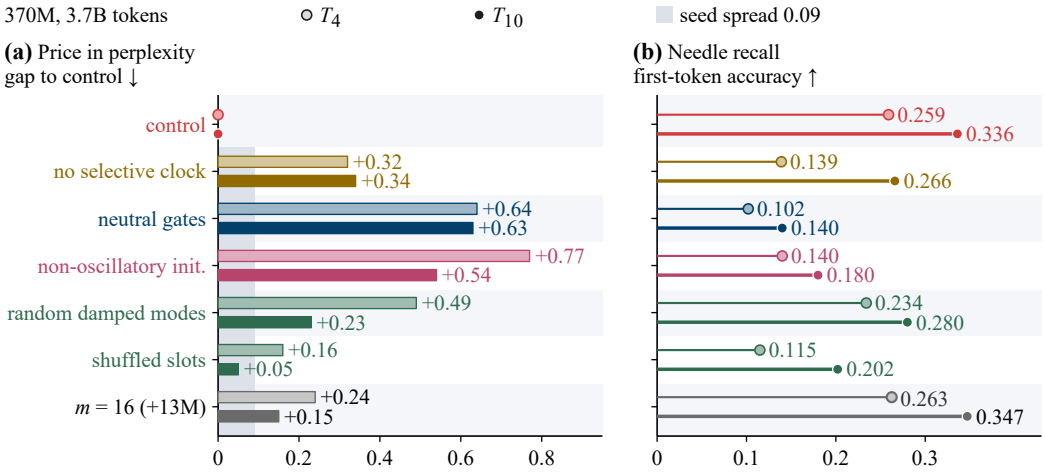

Figure 8: Effects of selective-mixer controls on language modeling and retrieval. (a) PPL changes relative to the control; positive values are worse, and shading shows the three-seed range of the control's full-capacity PPL (0.09; Table 29). (b) Needle first-token accuracy at full capacity and T_4. The $m = 16$ arm also increases parameter count and recurrent-state size. Table 27 gives the numerical values and checkpoint stages.

L.2 Real poles against rotating modes

We compare damped rotations with positive real poles when approximating the same Hankel filters. A rotation uses two real state coordinates, so eight rotations and sixteen real poles have equal state size (Table 28).

Eight rotations are more accurate on 14 of the 24 numerically resolved channels, including all eight channels retained by T_1. They fit eight channels within 5% error, versus six for sixteen real poles. Here, damped modes include zero-frequency terms. At channel eight the relative errors are 0.036 and 0.058, respectively. Sixteen real poles perform better on channels 9–17 and 20 and have the lower all-channel median, so the rotation fit's strongest advantage is on the low-index channels shared by every deployment tier.

The trained initialization controls provide complementary evidence. Non-oscillatory initialization raises PPL from 25.32 to 26.18 at T_1 and from 16.46 to 17.00 at full capacity, while random damped rotations reach 26.00 and 16.69 (Table 27). The non-oscillatory arm starts with identical decays $\rho = \mathrm{e}^{-0.01}$, zero frequencies and zero readout coefficients, giving zero initial mixer output. The

higher PPL under this control supports the fitted temporal initialization within the capacity-mixed training recipe.

Table 28: Relative filter-fit error at $L = 2048$ for positive real decays and damped rotations. Medians use the first 24 numerically resolved channels. Each damped mode uses two real state coordinates. Errors refer to the filter-fitting comparison on float64 Hankel eigenvectors.

	relative 2-norm error ↓					
Fit	median ($k \le 24$)	$k = 1$	$k = 4$	$k = 8$	$k = 16$	$k = 24$
rotations, $m = 8$ (initialization)	0.59	1.4×10^{-6}	2.2×10^{-4}	0.04	0.75	0.81
rotations, $m = 16$	8.9×10^{-6}	2.4×10^{-10}	7.8×10^{-8}	1.8×10^{-6}	8.8×10^{-7}	0.56
real poles, $m = 8$	0.76	9.2×10^{-3}	0.06	0.70	0.95	1.00
real poles, $m = 16$	0.30	4.2×10^{-4}	4.1×10^{-3}	0.06	0.54	0.95
real poles, $m = 32$	5.6×10^{-4}	3.9×10^{-7}	1.2×10^{-5}	2.6×10^{-4}	8.1×10^{-3}	0.11

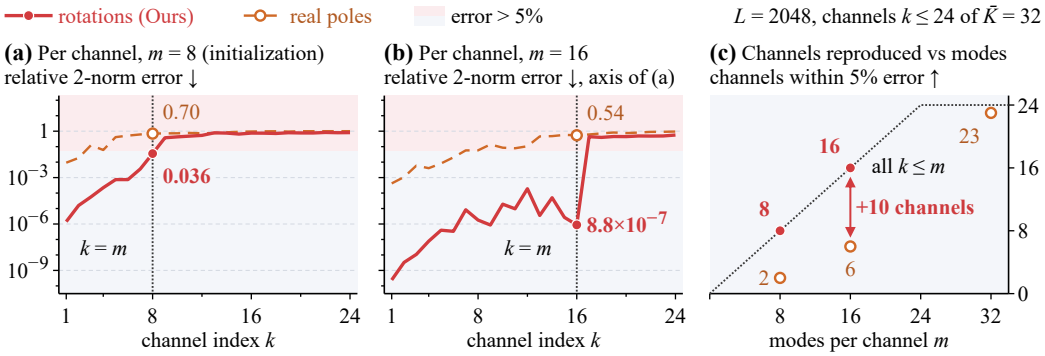

Figure 9: Finite-mode approximation of the Hankel spectral filters. Panels (a,b) show relative ℓ_2 error for eight and 16 modes over the first 24 numerically resolved channels; panel (c) counts channels below the stated error threshold as mode budget increases. A damped rotation uses two real state coordinates, while a positive real pole uses one. Panels (a,b) compare equal mode counts; eight rotations and sixteen real poles have equal state size. Table 28 reports the numerical summaries.

L.3 Initialization controls and the channel importance profile

The initialization controls vary either the temporal responses or the placement of complete channels. Random damped modes remove the fitted channel structure while preserving the initialization scale, whereas slot shuffling preserves each fitted channel but changes its position in the ordered bank. Uniform-decay and S4D-Lin (Gu et al., 2022) controls provide alternative starting timescales.

For the two seeds shared by the initialization controls, random damped modes give higher full and compact PPL than the Hankel initialization. Slot shuffling produces smaller PPL changes. The two S4D-Lin seeds give full-model PPL of 16.34–16.38, slightly below the Hankel control's 16.46–16.55 range. At reduced capacities, their T_4 PPL is approximately 19.61 and their T_1 range of 26.13–26.23 overlaps the Hankel control's 26.18–26.28 range. Thus both structured initializations support useful compact models under the same selective recurrence and capacity-mixed training procedure. Retrieval varies more strongly across seeds and is reported without ranking.

Knock-out importance is the validation-PPL increase after zeroing a channel, while output energy is its mean squared activation magnitude. Budget dropout concentrates predictive importance in the retained channels: the first eight account for about 64% of the knock-out effect in both control seeds, compared with 36% in the dense control. Their knock-out ordering is 0.75–0.82 versus 0.29 for the dense control. Random-mode and shuffled-slot models also develop strong ordering under budget dropout. Output energy is more seed dependent, with first-eight shares of 0.46 and 0.74 in the two controls, so it is reported separately from predictive importance.

Table 29: Initialization controls after 3.7B pretraining tokens, evaluated from raw checkpoints with the common frontier protocol. Full-key accuracy at T_4 is teacher forced. Rows report independent training seeds; retrieval varies substantially across seeds and is shown without ranking.

Initialization	Seed	PPL T_{10}	PPL T_4	PPL T_1	Full-key acc. T_4
Hankel control	197	16.46	19.59	26.19	0.249
Hankel control	313	16.55	19.67	26.28	0.030
Hankel control	511	16.52	19.64	26.18	0.134
Random damped modes	197	16.68	20.11	26.91	0.253
Random damped modes	313	16.84	20.26	27.14	0.177
Shuffled slots	197	16.51	19.73	26.49	0.129
Shuffled slots	313	16.50	19.72	26.28	0.164
S4D-Lin	197	16.34	19.61	26.13	0.202
S4D-Lin	313	16.38	19.61	26.23	0.205

Table 30: Initialization controls at 370M and 3.7B tokens. PPL differences use end-of-pretraining validation from the trainer (seed 197). Table 27 instead reports reduced-capacity PPL after shared refinement, while Table 29 uses the common PPL evaluator for independent seeds.

Arm	Change	ΔPPL $T_{10}\downarrow$	ΔPPL $T_1\downarrow$
ESSH (control)	–	0	0
ESSH (non-oscillatory init.)	identical real-decay modes, zero readout, no fit	$+0.54$	$+0.90$
ESSH (random damped modes)	fitted channel structure removed	$+0.23$	$+0.72$
ESSH (shuffled slots)	fitted channels placed in random slots	$+0.05$	$+0.30$
ESSH (reversed slots)	fitted channels placed in reverse order	$+0.07$	$+0.30$
ESSH (all decays $1-10^{-4}$)	uniform initial decay across modes	$+0.88$	$+1.21$
ESSH (all decays $1-10^{-3}$)	uniform initial decay across modes	$+0.89$	$+1.23$
ESSH (S4D-Lin init.)	S4D-Lin damped-mode initialization	-0.11	-0.05

Table 31: Channel importance at 370M after 3.7B tokens. Rows use seed 197 unless stated. Ordering is $-\mathrm{Spearman}(k, \mathrm{importance}_k)$, so larger values indicate a stronger decrease with channel index. Shares use the first eight channel indices. Output energy and knock-out importance are reported separately. The alternative map uses exponents $(1.0, 0.7)$; mixer-only nesting keeps FFN width fixed.

	Negative Spearman ↑		first-8 share	
Model	energy	knock-out	energy	knock-out
initialization (Hankel, untrained)	0.86	–	0.92	–
ESSH (control, seed 197)	0.66	0.82	0.46	0.64
ESSH (control, seed 313)	0.84	0.75	0.74	0.64
ESSH (alternative map)	0.74	0.70	0.51	0.57
ESSH (mixer-only nesting)	0.80	0.89	0.57	0.77
ESSH (dense control, nothing nested)	0.29	0.29	0.46	0.36
ESSH (non-oscillatory init.)	-0.05	0.88	0.37	0.45
ESSH (random damped modes)	0.15	0.87	0.85	0.75
ESSH (shuffled slots)	0.65	0.72	0.12	0.55
ESSH (all decays $1-10^{-4}$)	0.53	0.78	0.95	0.91
ESSH (all decays $1-10^{-3}$)	0.51	0.78	0.78	0.86

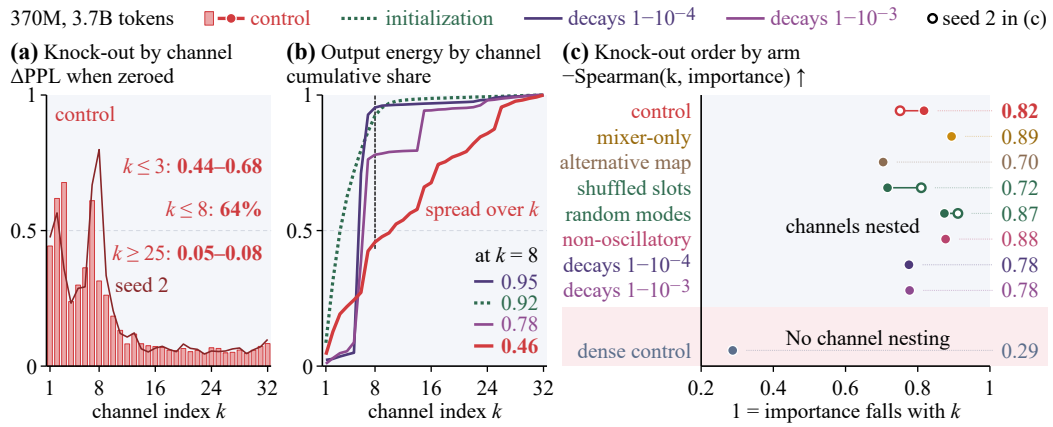

Figure 10: Concentration of predictive importance toward low-index spectral channels. (a) Knock-out importance measures the PPL increase after zeroing one channel. (b) Cumulative output energy measures activation magnitude and is distinct from knock-out importance. (c) Larger $-\,\mathrm{Spearman}(k, \mathrm{importance}_k)$ indicates a stronger decrease of importance with channel index. Open circles denote seed 313; panel (b) uses seed 197. Table 31 gives the corresponding statistics.

The intrinsic horizon $1/\gamma$ describes a mode's fixed decay, while a clock-adjusted attenuation statistic uses $\rho\,\mathbb{E}[\exp(-\Delta)]$. These are per-mode summaries. Retrieval measurements in Appendix H evaluate the complete hybrid, including its recurrent layers and shared attention.

L.4 Mode budget

Increasing the mode count improves the initial filter approximation but does not improve next-token loss in this sweep. With $m = 16$, PPL is higher at every reported tier than with $m = 8$, while needle first-token accuracy increases only slightly: from 0.259 to 0.263 at T_4 and from 0.336 to 0.347 at full capacity.

Each damped-rotation mode contributes two recurrent-state coordinates for every channel value. Doubling m from 8 to 16 therefore doubles recurrent-state storage at fixed channel shape and increases the full model from 370.9M to 384.0M parameters. We use $m = 8$ in the main recipe because it gives lower PPL at half the mode-state storage; $m = 16$ provides only the small retrieval gain shown above.

L.5 Channel budget

This sweep compares channel-budget configurations with $\bar{K}P$ fixed. The full recurrent-state size therefore remains

$$2\bar{K}mP = 2md.$$

Increasing $\bar{K}$ creates more independently controlled channel groups with fewer value coordinates per group, while also changing the number of selective-projection parameters. The sampler retains its fixed integer budget indices, so their fractions $\xi = c/\bar{K}$ change with $\bar{K}$. Shared refinement uses teacher index $c = 32$: this is full capacity for $\bar{K} \le 32$, but $\xi = 1/2$ for $\bar{K} = 64$, retaining 45 spectral channels under the two-rate map. The comparison therefore includes the corresponding training-capacity schedules.

The default $\bar{K} = 32$ gives the lowest full-capacity PPL among the elastic runs and is within 0.20 PPL of the best intermediate-tier values. Increasing to $\bar{K} = 64$ improves T_1 PPL to 23.46 but raises T_7 and full-capacity PPL to 17.78 and 16.60 and adds 13.1M parameters. We use $\bar{K} = 32$ to balance the full and reduced capacities within the 370M parameter budget. Retrieval values are single-seed measurements and are reported alongside the language-model results.

Table 32: Channel-count controls at 370M after 3.7B pretraining tokens. Channel value width P is adjusted so that the full recurrent-state size is constant. Compact elastic PPL uses final checkpoints; full PPL is measured from raw checkpoints. The retrieval column reports teacher-forced full-key accuracy. For both $\bar{K} = 64$ rows, retrieval uses $\xi = 1/2$ (45 retained spectral channels); the other retrieval values use full capacity. Dashes denote configurations without a reported evaluation; dense PPL entries use their full-capacity models.

		validation PPL ↓				full-key acc. ↑	params, full
Arm	P	T_1	T_4	T_7	T_{10}	NIAH	(M)
ESSH ($\bar{K} = 32$)	28	25.32	19.15	17.38	16.46	0.334	370.9
ESSH ($\bar{K} = 16$)	56	–	18.95	17.35	16.49	0.316	364.3
ESSH ($\bar{K} = 64$)	14	23.46	19.16	17.78	16.60	0.312	384.0
ESSH ($\bar{K} = 16$, dense control)	56	–	–	–	16.09	0.304	364.3
ESSH ($\bar{K} = 32$, dense control)	28	–	–	–	15.98	0.181	370.9
ESSH ($\bar{K} = 64$, dense control)	14	–	–	–	15.97	0.040	384.0

M Elasticity Ablations

These experiments isolate the choices that turn the nested tensor structure into a trained elastic family: the two-rate capacity map, FFN importance ordering, capacity sampling, gradient allocation, distillation, and the separation between recurrent and feed-forward capacity.

M.1 Nesting rule and ordering

The nesting sweep first asks which resources must shrink to obtain a materially smaller model (Table 33). Truncating spectral channels alone leaves the FFNs unchanged, so the smallest mixer-only export still contains 338.2M parameters, or 91.2% of the full model. Substantial parameter reduction therefore requires reducing FFN width as well as spectral-channel count. The two-rate map retains a larger fraction of spectral channels than FFN units while concentrating most parameter reduction in the FFNs.

The exponent sweeps show the resulting quality–size trade-off. Flatter channel or FFN reduction generally improves the smallest-tier PPL but also produces a larger export. For example, the flatter-FFN map retains 27.9% of the full parameters at T_1, compared with 24.8% for the control. No tested map lowers T_1 PPL at an equal or smaller export. The control also has the lowest full-capacity PPL among maps that reduce both axes, tied with $(1.0, 0.7)$. These results support the default exponents as a balance across capacities at the intended parameter budget.

At the same 92.1M shape, the recovered dense control has PPL 45.66 versus 25.32 for ESSH, supporting direct optimization of reduced capacities during training. FFN importance ordering provides a smaller, consistent improvement: removing it raises PPL at every measured tier by 0.07–0.37, whereas ordering every 1000 steps instead of every 500 changes PPL by at most 0.07. Training at reduced capacities accounts for the larger observed difference, while ordering improves the selection of retained FFN units.

M.2 Capacity sampling and optimization controls

The reserved full-capacity share f directly trades optimization between the full model and the reduced-capacity models. Reducing f from 0.75 to 0.5 improves T_1 PPL from 25.32 to 23.50 but increases full-capacity PPL from 16.46 to 17.01. Increasing f to 0.875 moves the endpoints in the opposite direction: full PPL improves to 16.26, while T_1 worsens to 27.56. Capacity-mixed training therefore allocates a finite update budget across predictors that share parameters but use different retained configurations.

Full-tier distillation primarily benefits the reduced-capacity models in this sweep. Removing it increases T_1 PPL from 25.32 to 26.42, while full PPL changes only from 16.46 to 16.48; T_4 needle first-token accuracy also decreases from 0.259 to 0.193. With coefficient 0.25, T_1 PPL is 25.92 and T_4 first-token accuracy is 0.290. Increasing the coefficient to 1.0 lowers T_1 PPL to 24.69 while

Table 33: Capacity-map and ordering controls. In the upper block, compact PPL uses final checkpoints and full PPL uses raw checkpoints. Different maps produce different parameter counts at the same nominal tier, so rows are not parameter matched. The lower diagnostic block reports the recorded T_4 needle first-token accuracy and raw T_1 PPL. For the dense control, the raw T_1 value is before compact recovery; the upper block reports the recovered compact checkpoint.

			validation PPL ↓			
Arm	exponents (channel, FFN)	r at T_1	T_1	T_4	T_7	T_{10}
ESSH (control)	$(0.5, 0.8)$	0.248	25.32	19.15	17.38	16.46
ESSH (alternative map)	$(1.0, 0.7)$	0.257	25.54	19.50	17.50	16.46
ESSH (steeper channels)	$(0.65, 0.8)$	0.237	26.34	19.40	17.50	16.50
ESSH (flatter channels)	$(0.35, 0.8)$	0.263	24.41	19.02	17.35	16.53
ESSH (flattest channels)	$(0.25, 0.8)$	0.278	23.74	18.76	17.26	16.47
ESSH (flatter feed-forward)	$(0.5, 0.7)$	0.279	23.89	18.81	17.25	16.52
ESSH (mixer-only nesting)	FFN fixed	0.912	16.82	16.22	16.10	16.12
ESSH (no FFN importance ordering)	$(0.5, 0.8)$	0.248	25.56	19.52	17.64	16.53
ESSH (ordering every 1000 steps)	$(0.5, 0.8)$	0.248	25.39	19.18	17.41	16.50
ESSH (dense control, reduced)	–	0.248	45.66	23.75	18.63	15.98

Arm	needle first-token acc. T_4 ↑	raw PPL T_1 ↓	params (M) at T_1
ESSH (control)	0.259	26.19	92.1
ESSH (alternative map)	0.152	26.42	95.3
ESSH (flatter channels)	0.331	25.26	97.6
ESSH (flattest channels)	0.220	24.54	103.0
ESSH (mixer-only nesting)	0.134	17.25	338.2
ESSH (no FFN importance ordering)	0.281	26.41	92.1
ESSH (dense control, reduced)	0.134	53162.13	92.1

the measured T_4 first-token accuracy is 0.055. We retain 0.5 to balance compact-model PPL and retrieval in this sweep.

The main recipe scales reduced-capacity FFN parameter gradients by $\alpha_{\text{ff}} = 0.5$. Here, α_{mix} denotes the analogous gradient scale applied to retained mixer parameters on reduced-capacity microbatches. The arm with $\alpha_{\text{ff}} = \alpha_{\text{mix}} = 0.5$ applies the same scaling to both retained FFN and mixer parameters. Both alternatives raise PPL at all four measured tiers: by 0.06–0.93 for $\alpha_{\text{ff}} = 0.25$, and by 0.04–0.40 when mixer gradients are also scaled. These results support scaling only the retained FFN gradients, at 0.5.

A weighted-label control tests whether increasing the CE coefficient reproduces the distillation result (Table 34). At T_1, increasing CE to $1.5\,\text{CE}$ lowers end-of-pretraining PPL from 27.61 to 26.76, while $\text{CE} + 0.5\,\text{KL}$ reaches 26.19. Thus label weighting recovers part of the improvement, while the distillation objective gives lower PPL at all four evaluated capacities in these runs.

Table 34: Reduced-capacity objective controls at 370M, 3.7B tokens and seed 197. All four PPL values use the end-of-pretraining model and the common 16-batch evaluator. Full-capacity microbatches always use CE. All three runs use 128 sequences per optimizer update.

Reduced-capacity objective	PPL T_1 ↓	PPL T_4 ↓	PPL T_7 ↓	PPL T_{10} ↓
CE	27.61	20.08	18.03	16.48
$1.5\,\text{CE}$	26.76	19.90	18.01	16.58
$\text{CE} + 0.5\,\text{KL}$	**26.19**	**19.59**	**17.83**	**16.46**

The learning-rate rows test sensitivity at 3.7B tokens. The 1.5×10^{-3} setting stays within 0.1 PPL of the control at every tier, while 2.0×10^{-3} gives lower PPL across the four tiers. The reported 1.5B run retains the configuration fixed before this study (Appendix Q.1).

Table 35: One-setting controls at 370M and 3.7B pretraining tokens. Compact PPL uses final checkpoints; full PPL and the seven-task mean use the corresponding full-model evaluations. The retrieval column reports NIAH first-token accuracy at the exact T_4 shape.

	validation PPL ↓				needle first-token acc. ↑	7-task ↑
Arm	T_1	T_4	T_7	T_{10}	T_4	T_{10}
ESSH (control)	25.32	19.15	17.38	16.46	0.259	0.408
ESSH ($f = 0.5$)	23.50	18.67	17.44	17.01	0.216	0.407
ESSH ($f = 0.875$)	27.56	19.92	17.53	16.26	0.199	0.410
ESSH ($\alpha_{\text{ff}} = 0.25$)	26.25	19.59	17.59	16.52	0.135	0.410
ESSH ($\alpha_{\text{ff}} = \alpha_{\text{mix}} = 0.5$)	25.72	19.31	17.48	16.50	0.162	0.413
ESSH (no full-tier distillation)	26.42	19.53	17.51	16.48	0.193	0.412
ESSH (KL coefficient 0.25)	25.92	19.35	17.45	16.49	0.290	0.406
ESSH (KL coefficient 1.0)	24.69	18.96	17.35	16.53	0.055	0.413
ESSH (lr 1.5×10^{-3})	25.23	19.08	17.32	16.47	0.150	0.406
ESSH (lr 2.0×10^{-3})	24.99	18.90	17.15	16.34	0.309	0.409

The sampling schedule determines how often the reduced-capacity models are optimized directly. Sampling only five deployment tiers, $T_1, T_3, T_5, T_7, T_{10}$, leaves the omitted T_2 with the largest measured penalty, $+2.59$ PPL, despite sampling both family endpoints. Uniformly sampling all ten deployment tiers instead improves some larger tiers but worsens T_1 and T_2 by 1.38 and 1.37 PPL relative to the control menu. These results show that endpoint coverage alone is insufficient and that the set of training capacities and deployment grid serve different roles.

The progressive schedule initially samples training capacities $\{16, 24\}/32$, then adds $\{8, 12\}/32$, $\{4, 5, 6\}/32$, and $\{2, 3\}/32$ at 15%, 30%, and 45% of training, respectively. The share ramp increases the reserved full-capacity fraction linearly from 0.75 to 1.0 over the second half. Gap-proportional sampling updates every 500 steps using each reduced capacity's nonnegative validation cross-entropy gap to full capacity, with linear interpolation in $\log_2 K$ for unevaluated capacities. It uses full capacity until the initial validation at step 0, then samples reduced capacities using the resulting weights.

The adaptive-share variant also updates every 500 steps and activates its weights after the second validation, at step 500, once loss changes are available. In both weighted-sampling variants, activation bypasses the remainder of the default 250M-token full-capacity warm-up. It assigns larger exponential weights to capacities whose validation loss has decreased less over the preceding interval, using a temperature equal to half the mean absolute loss decrease. The resulting full-capacity share is clipped to $[0.5, 0.9]$ and changes by at most 0.05 per subsequent update. Reduced-capacity weights are interpolated in $\log_2 K$ and mixed equally with uniform sampling. Compared with the fixed-share control, the ramp improves full PPL by 0.14 but worsens T_1 by 2.34; gap-proportional sampling reverses that emphasis, improving T_1 by 1.66 while worsening full PPL by 0.21. Adaptation therefore redistributes quality across the family without improving all measured tiers together.

Table 36: Raw PPL differences relative to the fixed-share control for alternative 370M sampling schedules after 3.7B tokens. Negative values indicate lower PPL than the control. Columns evaluate the exact deployment shapes.

Schedule	T_1	T_2	T_4	T_7	T_{10}
ESSH (fixed share $f = 0.75$, control)	0	0	0	0	0
ESSH (fixed share $f = 0.5$)	-2.16	-1.30	-0.58	-0.02	$+0.55$
ESSH (adaptive share, 0.5 to 0.9)	-2.59	-0.96	-0.25	$+0.17$	$+0.52$
ESSH (sampled capacities = the ten tiers)	$+1.38$	$+1.37$	$+0.01$	-0.28	-0.08
ESSH (sampled capacities = five tiers $T_1, T_3, T_5, T_7, T_{10}$)	$+0.10$	$+2.59$	$+0.23$	-0.21	-0.03
ESSH (progressive capacity reduction, four stages)	$+0.92$	$+0.11$	-0.10	-0.05	$+0.07$
ESSH (share ramp 0.75 to 1.0 over the second half)	$+2.34$	$+1.47$	$+0.79$	$+0.22$	-0.14
ESSH (gap-proportional sampling)	-1.66	-0.48	$+0.14$	$+0.33$	$+0.21$

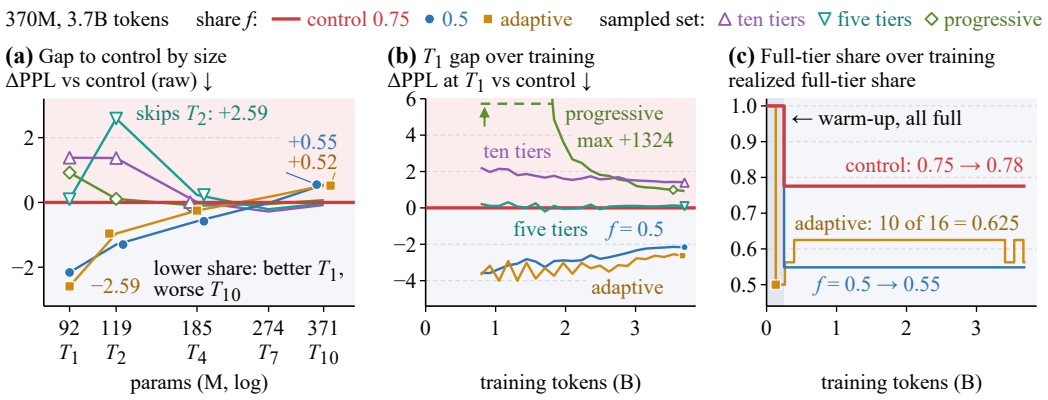

Figure 11: Selected capacity-sampling schedules. Panel (a) shows PPL differences at the five evaluated tiers, panel (b) tracks T_1 during training, and panel (c) shows the realized full-capacity share. The progressive schedule's early maximum of +1324 PPL is clipped by the plotting range. The complete schedule comparison, including the share ramp and gap-proportional variant, is in Table 36.

M.3 Full-capacity cost of shared training

The dense ESSH control isolates the full-capacity PPL cost of optimizing multiple capacities with shared parameters. At matched architecture and pretraining budget, the raw full-capacity PPL gap between elastic and dense training is $+0.48$ after 3.7B tokens and $+0.41$ after 7.4B tokens (Table 37). These gaps are larger than the observed seed ranges for either training condition. The independently trained Mamba-3 + SWA reference gives PPL 14.10 and 13.97 for seeds 197 and 313 under the same frontier protocol, a range of 0.13. The second run reproduces the reference model's lower full-capacity PPL.

This cost is consistent with the allocation trade-off in the full-share sweep: increasing full-capacity participation improves the full endpoint while weakening the smallest export. The resulting reduced-capacity models benefit from those shared updates: at equal total training exposure, T_4 PPL is 19.15 for ESSH versus 23.75 for the truncated-and-recovered dense control (Table 33). Shared refinement subsequently improves the delivered elastic family and is reported separately with its own token budget and checkpoint stage.

Table 37: Full-capacity PPL at the end of pretraining. ESSH comparisons use the same architecture and token budget within each block. Mamba-3 + SWA is a separate independently trained reference. PPL reports seed 197; range is the maximum minus the minimum across three seeds for 3.7B elastic ESSH and two for each other training condition.

Model (370M scale)	tokens (B)	PPL T_{10} ↓	seed range
ESSH	3.7	16.46	0.09
ESSH (dense control, no capacity mixing)	3.7	15.98	0.06
PPL gap (elastic − dense)	3.7	+0.48	–
ESSH	7.4	14.85	0.05
ESSH (dense control, no capacity mixing)	7.4	14.44	0.05
PPL gap (elastic − dense)	7.4	+0.41	–
Independent full-model reference			
Mamba-3 + SWA (377.1M)	7.4	14.10	0.13

M.4 Mixer and feed-forward budgets

Spectral channels, modes, and FFN width allocate different resources. Channels partition the recurrent representation, modes determine the temporal-state budget, and the FFN provides tokenwise

nonlinear capacity. Table 38 varies FFN width while keeping depth, residual width, and the spectral-state configuration fixed.

Increasing FFN width improves PPL at every reported tier and also increases T_4 needle first-token accuracy. The marginal full-capacity PPL gain nevertheless decreases across the sweep. Increasing d_{ff} from 3072 to 4608 adds 90.83M full-model parameters and improves full PPL by 0.66; increasing it from 4608 to 6144 adds the same 90.83M parameters but improves full PPL by 0.37.

Keeping this tokenwise expansion outside the recurrent mixer makes the two budgets independently adjustable: increasing FFN width changes parameters and tokenwise computation without multiplying recurrent-state storage. The two-rate map uses the same separation in the opposite direction, reducing the parameter-heavy FFN faster than the spectral-channel count. We use $d_{\text{ff}} = 4608$ to meet the 370M parameter budget used for the baseline comparison (Appendix E.6).

Table 38: FFN-width sweep at fixed depth, residual width, and spectral mixer after 3.7B pretraining tokens. Exported parameter counts vary with FFN width. Compact PPL uses final checkpoints; full-capacity PPL uses the corresponding raw checkpoints.

d_{ff}	params T_1/T_{10} (M)	PPL $T_1 \downarrow$	PPL $T_4 \downarrow$	PPL $T_7 \downarrow$	PPL $T_{10} \downarrow$	needle first-token acc. $T_4 \uparrow$
3072	84.6 / 280.0	26.44	20.26	18.29	17.12	0.191
4608 (ESSH)	92.1 / 370.9	25.32	19.15	17.38	16.46	0.259
6144	103.5 / 461.7	24.18	18.59	16.91	16.09	0.319

N Compression and Train-Once Baselines

We compare three ways to obtain multiple deployment sizes: joint training of nested predictors, post-hoc recovery from a pretrained parent, and independent training at each target shape. Because these procedures use different training stages and token budgets, we report training exposure, recovery steps, and actual exported parameter counts separately.

N.1 Elastic-family comparison curves

The family comparison evaluates ESSH, MatFormer and MatMamba using actual exported parameter counts (Figure 3(a)). MatFormer varies feed-forward width, while MatMamba varies mixer inner width and the corresponding head count (Devvrit et al., 2024; Shukla et al., 2024). The local baselines use four training capacities: $1/8$, $1/4$, $1/2$, and full width. The local MatMamba implementation reserves one quarter of micro-batches for full width and samples the other three widths uniformly in the remaining micro-batches, forming a stochastic version of the nested-width recipe.

The baseline rows at the fractions assigned to T_2 and T_{10} coincide with trained capacities, while T_3–T_9 are interpolated and T_1 is below the trained range. ESSH also evaluates interpolated capacities: its 370M T_3–T_7 and T_9 are not sampled directly during training. The comparisons therefore assess both directly trained and intermediate exports, with the below-range baseline results separated in the tables.

ESSH uses its final shared model, while MatFormer and MatMamba use their 7.4B-token models without an additional shared stage. Before that stage, ESSH already has PPL 17.94 at T_4, 16.22 at T_7, and 14.85 at full capacity. These are below the comparable 188.4M MatMamba result of 18.47, the 270.9M MatFormer result of 16.98, and the two full-baseline results of 16.80–16.98. Final ESSH further reaches 16.98 at the 214.5M T_5.

The baselines' additional directly trained widths provide useful references: MatFormer has PPL 17.93 at 176.3M and 17.03 at 240.7M, and MatMamba has 19.21 at 140.6M and 17.50 at 217.1M. These values support the intermediate-size comparison without relying solely on interpolated baseline configurations. The ΔNLL columns equal $\ln(\text{PPL}_T/\text{PPL}_{T_{10}})$ within each family; absolute PPL and parameter count determine cross-family quality comparisons.

The four independent Transformer++ models test quality at similar deployment sizes (Table 41). ESSH has lower PPL and higher seven-task mean at T_4, T_7, and T_{10}; the independently trained

Table 39: MatFormer at the ten ESSH deployment budget fractions, using its end-of-pretraining 7.4B-token model. Architecture-specific retained dimensions and parameter counts differ from ESSH. r is the retained parameter fraction; ΔNLL is relative to the full model. T_1 lies below the smallest capacity used during MatFormer training. Seed 197. Latency is a batch-one CUDA-graph measurement after a 2K context; see Appendix E.5.

Tier	Params (M)	r	PPL ↓	ΔNLL ↓	7-task ↑	ms/token ↓
T_2	146.0	0.400	19.37	0.142	0.390	3.14
T_3	168.7	0.462	18.48	0.095	0.396	3.21
T_4	191.5	0.524	17.82	0.058	0.401	3.29
T_5	214.2	0.586	17.50	0.041	0.404	3.27
T_6	248.2	0.679	17.02	0.012	0.408	3.34
T_7	270.9	0.741	16.98	0.011	0.410	3.37
T_8	301.2	0.824	16.93	0.007	0.410	3.39
T_9	331.5	0.907	16.87	0.004	0.411	3.44
T_{10}	365.6	1.000	16.80	0.000	0.411	3.46
Below the smallest trained capacity						
T_1	130.9	0.358	5844.15	5.852	0.282	3.02

Table 40: MatMamba at the ten ESSH deployment budget fractions, using its end-of-pretraining 7.4B-token model. Architecture-specific retained dimensions and parameter counts differ from ESSH. r is the retained parameter fraction; ΔNLL is relative to the full model. T_1 lies below the smallest capacity used during MatMamba training. Seed 197. Latency is a batch-one CUDA-graph measurement after a 2K context; see Appendix E.5.

Tier	Params (M)	r	PPL ↓	ΔNLL ↓	7-task ↑	ms/token ↓
T_2	102.4	0.277	22.06	0.262	0.371	3.21
T_3	131.1	0.354	20.53	0.190	0.379	3.03
T_4	159.7	0.432	19.10	0.118	0.386	3.19
T_5	188.4	0.509	18.47	0.084	0.390	3.18
T_6	226.6	0.613	17.47	0.028	0.398	3.24
T_7	255.3	0.690	17.38	0.023	0.400	3.36
T_8	293.5	0.793	17.24	0.015	0.399	3.44
T_9	331.7	0.897	17.11	0.008	0.403	3.44
T_{10}	369.9	1.000	16.98	0.000	0.404	3.42
Below the smallest trained capacity						
T_1	83.3	0.225	3029.37	5.184	0.275	2.99

smallest model is stronger on both metrics at T_1. For example, the near-274M comparison gives PPL 15.91 versus 16.69 and task mean 42.00% versus 40.79%. Shared training therefore retains both next-token and task quality at the intermediate sizes, with the largest sharing cost at the smallest export.

Table 41: PPL and seven-task mean accuracy (%) at four deployment sizes. ESSH follows the complete shared-training procedure; Transformer++ uses four independent runs. Both use 7.4B pretraining tokens with the same data, tokenizer and sequence length. ESSH's additional shared stage is specified in Appendix E.3.

	Params (M)		PPL ↓		7-task ↑	
Target	ESSH	TF++	ESSH	TF++	ESSH	TF++
T_1	92.1	92.7	23.61	20.36	37.56	38.41
T_4	185.3	186.3	17.68	17.76	40.45	39.84
T_7	274.3	273.6	15.91	16.69	42.00	40.79
T_{10}	370.9	365.6	14.71	15.88	42.84	41.38

N.2 Independent training seeds

The seed comparison asks whether the intermediate-size PPL differences exceed the observed variation across independent runs of each family. Near 274M parameters, the higher of the two ESSH values is 15.96, still below the lowest MatFormer value of 16.98. Full-capacity ESSH likewise remains below the completed MatFormer and MatMamba runs in this recipe-level comparison.

The baseline T_1 results are much more variable because this tier lies below the capacities sampled during baseline training. We therefore report those values but exclude them from the common-range interpretation.

Table 42: PPL across independent training seeds at four deployment shapes. ESSH uses final shared checkpoints; MatFormer and MatMamba use end-of-pretraining checkpoints. Baseline T_1 lies below the smallest capacity sampled during their training and is not used for the common-range comparison.

Family	Seed	PPL T_1 ↓	PPL T_4 ↓	PPL T_7 ↓	PPL T_{10} ↓
ESSH	197	23.61	17.68	15.91	14.71
ESSH	511	23.77	17.76	15.96	14.73
MatFormer	197	5844.15	17.82	16.98	16.80
MatFormer	313	106.56	18.08	17.30	17.14
MatFormer	511	112.01	18.05	17.25	17.10
MatMamba	197	3029.37	19.10	17.38	16.98
MatMamba	313	13651.22	19.12	17.41	17.01

N.3 Shape-matched recovery and specialist references

Table 43 compares ways to obtain the exact T_4 and T_7 deployment shapes. The first block starts from 3.7B pretraining tokens and contrasts capacity-mixed training with truncating and recovering a full-capacity-only model. The second block compares the 7.4B shared family with specialists trained independently at each target size.

Within the 3.7B block, ESSH reaches full-key accuracy 0.258 at T_4 and 0.256 at T_7, compared with at most 0.146 and 0.116 for the recovered dense controls. It also has lower T_4 PPL, while 4120-step label-only recovery reaches similar T_7 PPL with twice the recovery steps. In the 7.4B block, the specialists have lower PPL and slightly higher task means, while ESSH retains higher full-key accuracy and supplies both sizes from shared training.

Table 43: Models at the exact ESSH T_4/T_7 shapes, grouped by pretraining budget. Additional tokens and steps specify the subsequent shared-training or recovery stage. Full-key accuracy uses the common 2K NIAH protocol.

	Additional training		T_4, 185.3M			T_7, 274.3M		
Method	Tokens (B)	Steps	PPL ↓	7-task ↑	Full key ↑	PPL ↓	7-task ↑	Full key ↑
3.7B pretraining tokens								
ESSH	0.27	2060	19.15	0.394	0.258	17.38	0.401	0.256
Dense control, truncated and recovered	0.27	2060	23.75	0.376	0.131	18.63	0.397	0.116
Dense control, label-only recovery	0.54	4120	20.07	0.386	0.146	17.22	0.407	0.114
7.4B pretraining tokens								
ESSH	0.27	2060	17.68	0.405	0.370	15.91	0.420	0.366
Independent specialist$^\diamond$	0	–	16.25	0.412	0.326	15.10	0.424	0.306

N.4 Post-hoc elastification of the reference hybrid

The post-hoc comparison recovers the Mamba-3 + SWA parent into models near the ESSH T_4 and T_7 sizes. We compare seven-task accuracy and full-key retrieval under the common evaluation protocols.

At the larger compact target, the two models have similar seven-task means: 0.425 for the post-hoc model and 0.420 for ESSH. Their teacher-forced full-key accuracies differ more substantially, 0.147 and 0.366, respectively. At the smaller target, ESSH is higher on both common metrics, with seven-task mean 0.405 versus 0.397 and full-key accuracy 0.370 versus 0.034. The compact models can therefore have similar aggregate task accuracy while differing markedly on planted-key retrieval.

Table 44: Post-hoc masks targeting T_4/T_7 sizes. Recovered Mamba-3 models use 7.90B total tokens; ESSH uses 7.67B. Seven-task and full-key metrics use the same evaluation protocols. Actual parameter counts are shown.

Configuration	tokens (B)	params (M)	7-task mean ↑	Full-key acc. ↑
parent, dense Mamba-3 + SWA*	7.4	377.1	0.445	0.230
100% mask, after elastification	7.9	377.1	0.438	0.248
mask targeting T_4	7.9	185.0	0.397	0.034
ESSH, T_4	7.67	185.3	0.405	0.370
mask targeting T_7	7.9	273.8	0.425	0.147
ESSH, T_7	7.67	274.3	0.420	0.366

N.5 Preserving attention in post-hoc masks

The attention-preserving control uses the same parent and 0.5B-token recovery budget while preserving all heads in both attention layers; the feed-forward units within those blocks remain subject to pruning. At the T_7 target, this changes full-key accuracy only from 0.147 to 0.148, compared with 0.366 for the near-size ESSH export. The T_4 attention-preserving mask likewise remains far below ESSH.

Retaining the attention components alone is therefore insufficient to recover the retrieval behavior under this post-hoc recipe. The result is consistent with the broader comparison above: jointly training the reduced-capacity predictor produces different behavior from first training a dense model and recovering its compact masks afterwards.

N.6 Post-hoc elastification of the dense control

This comparison fixes the ESSH architecture, the 3.7B-token parent pretraining budget, and the exported T_4/T_7 shapes, while changing how the compact models are trained. Post-hoc elastification

Table 45: Original and attention-preserving post-hoc masks near the T_4/T_7 sizes. Both recovery strategies use the same parent and 0.5B recovery tokens. Task and retrieval scores use the common protocols.

Model / target	Params (M)	First token ↑	Full key ↑	7-task ↑
Mamba-3 + SWA parent	377.1	0.241	0.230	0.445
Original mask, T_4 target	185.0	0.036	0.034	0.397
Preserve attention, T_4 target	185.3	0.025	0.023	0.395
ESSH, T_4	185.3	0.370	0.370	0.405
Original mask, T_7 target	273.8	0.152	0.147	0.425
Preserve attention, T_7 target	276.5	0.154	0.148	0.426
ESSH, T_7	274.3	0.366	0.366	0.420

adds 0.5B recovery tokens for 4.20B total tokens; capacity-mixed training is followed by 0.27B shared refinement for 3.97B total tokens.

At T_7, post-hoc recovery and ESSH have nearly identical PPL, 17.35 and 17.38, but full-key accuracy is 0.107 and 0.256, respectively. At T_4, both have seven-task mean 0.394, while full-key accuracy is 0.154 after post-hoc recovery and 0.258 for ESSH. Thus, under the same architecture and deployment shape, similar aggregate PPL or task accuracy can coexist with substantially different retrieval behavior.

These controls motivate training the retained configurations throughout pretraining rather than relying only on post-hoc recovery. Under the measured recipes, capacity-mixed training preserves more of the planted-key behavior despite using fewer total training tokens.

Table 46: Same-architecture compact-model comparison. Post-hoc elastification uses 4.20B total tokens; capacity-mixed ESSH and the directly reduced dense control use 3.97B. Compact rows use the exact deployment shapes. Full-key is teacher-forced full-key accuracy.

Configuration	tokens (B)	params (M)	PPL ↓	7-task mean ↑	Full-key acc. ↑
parent: ESSH (dense control, 3.7B)	3.7	370.9	15.98	0.414	0.181
100% mask, after elastification	4.2	370.9	16.15	0.417	0.113
Post-hoc T_4	4.2	185.3	19.64	0.394	0.154
ESSH (3.7B), T_4	3.97	185.3	19.15	0.394	0.258
ESSH (dense control, reduced), T_4	3.97	185.3	23.75	0.376	0.131
Post-hoc T_7	4.2	274.3	17.35	0.404	0.107
ESSH (3.7B), T_7	3.97	274.3	17.38	0.401	0.256
ESSH (dense control, reduced), T_7	3.97	274.3	18.63	0.397	0.116

N.7 Training cost within the ESSH architecture

The internal cost comparison asks how much accelerator time is required to obtain the same four ESSH deployment shapes, $T_1/T_4/T_7/T_{10}$, either from one shared training run or from four independent runs. Pretraining A100-hours are estimated by dividing 7.4B tokens by each configuration's rate over steps 20–35 of a single-A100 probe. The shared run includes capacity mixing and teacher forwards; shared refinement uses its measured 2060-step elapsed time. These estimates exclude restarts, evaluation, and data-loading stalls.

The shared procedure requires an estimated 53.9 A100-hours under this accounting, compared with 50.6 hours for one independently trained full model and 147.4 hours for four independent models. Thus producing the four-size family reduces the four-model total from 147.4 to 53.9 A100-hours, a 63.4% reduction, while requiring only about 6.5% more than the single full-model run.

The shared family obtains this saving while staying within 0.009 of the independent models on seven-task mean at all four shapes. Independent training gives lower compact-model PPL, with the largest difference at T_1 (Table 11); the cost comparison accounts for obtaining all four sizes.

Table 47: Internal ESSH cost comparison for obtaining $T_1/T_4/T_7/T_{10}$ on one A100-80GB. Pretraining A100-hours are computed from 7.4B tokens and measured steady-state throughput; shared refinement uses the measured 2060-step wall-clock time. The four deployment sizes are 92.1, 185.3, 274.3, and 370.9M parameters.

Stage	Tokens (B)	ktok/s	A100-h ↓
Shared pretraining	7.4	41.9	49.1
Shared refinement	0.27	–	4.81
Independent T_1	7.4	90.2	22.8
Independent T_4	7.4	62.1	33.1
Independent T_7	7.4	50.2	40.9
Independent T_{10}	7.4	40.66	50.6
Shared training, total			53.9
Four independent models, total			147.4

N.8 Training cost of the external four-model comparison

The four independently trained Transformer++ models require an estimated 101.4 A100-hours to obtain the four measured deployment sizes. Shared ESSH requires 53.9 hours, including its shared refinement stage, reducing the total by 46.9% (Table 48). The quality comparison uses these same four sizes; their task accuracies are listed in Table 41.

The independent total credits the different training rates of the four models: the smallest requires 14.55 hours, compared with 35.52 for the full-size reference. The saving therefore persists after accounting for the lower cost of each compact model. Shared training reuses updates to the retained tensors across deployment tiers, while the independent family optimizes four separate parameter sets.

The full-size and smallest Transformer++ rates come from completed single-A100 training segments. The two intermediate rates come from matched 100-step probes with the same frozen architecture, global token batch, optimizer, and execution settings. We discard the first 20 steps and use the median of seven ten-step blocks over steps 20–90, reducing the effect of 0.1-second timestamp resolution. The median rates are 91.98 and 70.85 ktok/s for the T_4 and T_7 targets, respectively. Shared ESSH accounting is defined in Appendix N.7.

Table 48: Training cost for the four sizes in Table 41. All rates use one A100-80GB. Independent costs are estimated as 7.4B tokens divided by each model's steady-state rate. Shared ESSH includes the teacher computation and measured shared refinement stage. The reported costs are estimates of training work at the measured rates.

Transformer++ target / total	Rate (ktok/s)	A100-hours
T_1	141.32	14.55
T_4	91.98	22.35
T_7	70.85	29.01
T_{10}	57.87	35.52
Four independent Transformer++ runs	–	101.43
One shared ESSH run	–	**53.90**

O Systems: Algorithms and Kernels

This section gives the two execution paths used by ESSH: a chunked parallel scan for training and a tokenwise recurrent step for decoding. Both evaluate the same selective recurrence, including the channel-local value blend, zero-lag correction, and tier-specific tensor slices.

O.1 CHUNKED FORWARD SCAN

The chunk decomposition follows the structured state-space duality view of sequence computation (Dao & Gu, 2024), specialized to the retained selective spectral channels.

For channel k and mode i, let $\psi_{k,i}(\tau)$ denote the damped-mode response in Equation 12. The single-mode causal interaction is

$$G_{k,i}(t,s) = \mathbf{1}[s \le t]\, \eta^{\mathrm{r}}_{k,i}(t)\eta^{\mathrm{w}}_{k,i}(s)\mathrm{e}^{-(q_{k,i}(t)-q_{k,i}(s))}\psi_{k,i}(t-s). \tag{27}$$

Entries above the causal diagonal are zero and need not be evaluated.

Consider a chunk $[b,e]$. Let

$$G_k[t,s] = \sum_{i=1}^{m} G_{k,i}(t,s),$$

and stack the channel values as

$$V_k = [v_k(b)^\top; \ldots; v_k(e)^\top].$$

The contribution from the state entering the chunk is

$$B_k[t,:] = \sum_{i=1}^{m} \eta^{\mathrm{r}}_{k,i}(t)\kappa^{\top}_{k,i}\Pi_{k,i}(t,b-1)H_{k,i}(b-1), \qquad b \le t \le e.$$

Thus

$$G_k \in \mathbb{R}^{(e-b+1)\times(e-b+1)}, \qquad V_k, B_k, Z^{\mathrm{core}}_k \in \mathbb{R}^{(e-b+1)\times P}.$$

Unrolling from the entering state gives

$$\begin{aligned} H_{k,i}(t) &= \Pi_{k,i}(t,b-1)H_{k,i}(b-1) \\ &\quad + \sum_{s=b}^{t} \Pi_{k,i}(t,s)\eta^{\mathrm{w}}_{k,i}(s)e_1 v_k(s)^\top, \\ Z^{\mathrm{core}}_k &= \left(\sum_i G_{k,i}\right) V_k + B_k = G_k V_k + B_k. \end{aligned} \tag{28}$$

The state passed to the next chunk is

$$H_{k,i}(e) = \Pi_{k,i}(e,b-1)H_{k,i}(b-1) + \sum_{s=b}^{e} \Pi_{k,i}(e,s)\eta^{\mathrm{w}}_{k,i}(s)e_1 v_k(s)^\top.$$

The local injection at $s=b$ does not include $A_{k,i}(b)$, whereas the entering state does. Equation 28 therefore accounts for the boundary without double counting. Repeating the update over chunks reproduces the tokenwise recurrence, including a final partial chunk. Equation 24 is then applied once at each valid position.

Interaction across a chunk boundary. Fix the realized controls. For a source $s < b$ and target $t \ge b$,

$$G_{k,i}(t,s) = \left[\eta^{\mathrm{r}}_{k,i}(t)\kappa^{\top}_{k,i}\Pi_{k,i}(t,b-1)\right]\left[\Pi_{k,i}(b-1,s)\eta^{\mathrm{w}}_{k,i}(s)e_1\right]. \tag{29}$$

Each mode therefore contributes a token-interaction factorization of inner dimension two. Concatenating the factors over m modes gives rank at most $2m$ for the scalar token-interaction block crossing the boundary. The zero-lag correction contributes no cross-boundary term. With P value coordinates, the corresponding recurrent state contains $2mP$ entries per channel. This rank statement applies to the cross-boundary interaction block, not to the complete causal matrix.

Shared values across modes. All modes in channel k act on the same V_k, so

$$\sum_i G_{k,i}V_k = \left(\sum_i G_{k,i}\right)V_k.$$

The mode tiles can therefore be accumulated before the value multiplication. For a full Q-token chunk, the recurrent core requires

$$O(K_T m Q^2) + O(K_T Q^2 P) + O(K_T Q m P)$$

work for tile construction, shared-value multiplication, and boundary propagation. Across a length-L sequence this becomes

$$O(L K_T [Q(m+P) + mP]),$$

up to padding of the final chunk. Projection costs are accounted for separately in Table 5.

Algorithms 2 and 3 use

$$\Pi_{k,i}(t,s) = A_{k,i}(t) \cdots A_{k,i}(s+1), \qquad \Pi_{k,i}(t,t) = I_2.$$

H^{bd} denotes the state entering a chunk, and $\mathcal{H}$ stores these boundary states for backward evaluation. Within chunk j, the local accumulated clock

$$q^{(j)}_{k,i}(t) = \sum_{s=b_j}^{t} \Delta_{k,i}(s)$$

produces the same within-chunk differences $q^{(j)}(t) - q^{(j)}(s) = q(t) - q(s)$. Batch indices are omitted.

Algorithm 2: Chunked forward evaluation

Input: $v_k(t), \eta^{\mathrm{w}}_{k,i}(t), \eta^{\mathrm{r}}_{k,i}(t), \Delta_{k,i}(t), \rho, \theta, \kappa, a, L, Q, K_T$, flag save
Output: corrected channel outputs $z_k(1{:}L)$ and boundary cache $\mathcal{H}$

$\mathcal{H} \leftarrow \varnothing$
foreach $k = 1, \ldots, K_T$ *in parallel* **do**
  $H^{\mathrm{bd}}_{k,i} \leftarrow 0$ for $i = 1, \ldots, m$
  for $j = 1, \ldots, \lceil L/Q \rceil$ **do**
    $b \leftarrow (j-1)Q + 1$
    $e \leftarrow \min(jQ, L)$
    $\mathcal{C}_j \leftarrow \{b, \ldots, e\}$
    if save **then**
      $\mathcal{H}_{k,:,j} \leftarrow H^{\mathrm{bd}}_{k,:}$
    $V_k \leftarrow [v_k(b)^\top; \ldots; v_k(e)^\top]$
    $q^{(j)}_{k,i}(t) \leftarrow \sum_{s=b}^{t} \Delta_{k,i}(s)$ for each (i,t) with $t \in \mathcal{C}_j$
    $G_k[t-b+1, s-b+1] \leftarrow \sum_{i=1}^{m} G_{k,i}(t,s)$ using $q^{(j)}$ for $t, s \in \mathcal{C}_j$
    if $e - b + 1 < Q$ **then**
      $(G_k, V_k) \leftarrow \mathrm{ZeroPad}(G_k, V_k; Q)$
    $Z^{\mathrm{local}}_k \leftarrow G_k V_k$
    foreach $t \in \mathcal{C}_j$ *in parallel* **do**
      $z^{\mathrm{core}}_k(t) \leftarrow (Z^{\mathrm{local}}_k[t-b+1, :])^\top$
      $z^{\mathrm{core}}_k(t) \leftarrow z^{\mathrm{core}}_k(t) + \sum_i \eta^{\mathrm{r}}_{k,i}(t) (H^{\mathrm{bd}}_{k,i})^\top \Pi_{k,i}(t, b-1)^\top \kappa_{k,i}$
      $z_k(t) \leftarrow z^{\mathrm{core}}_k(t) - \sum_i a_{k,i} \eta^{\mathrm{r}}_{k,i}(t) \eta^{\mathrm{w}}_{k,i}(t) \kappa^{\mathrm{c}}_{k,i} v_k(t)$
    foreach $i = 1, \ldots, m$ *in parallel* **do**
      $H^{\mathrm{bd}}_{k,i} \leftarrow \Pi_{k,i}(e, b-1) H^{\mathrm{bd}}_{k,i}$
      $H^{\mathrm{bd}}_{k,i} \leftarrow H^{\mathrm{bd}}_{k,i} + \sum_{s=b}^{e} \Pi_{k,i}(e,s) \eta^{\mathrm{w}}_{k,i}(s) e_1 v_k(s)^\top$
return $z, \mathcal{H}$

O.2 Backward pass with boundary adjoints

The output projection gives the upstream channel gradient

$$g_k(t) = K_T^{-1/2} (W^{\mathrm{out}}_k)^\top \nabla_{\hat{y}^{(T)}(t)} \mathcal{L}.$$

Fix one channel and suppress its index. For one mode, define the state adjoint backward from the final position:

$$E_L = \eta^{\mathrm{r}}_L \kappa g_L^\top, \qquad E_t = \eta^{\mathrm{r}}_t \kappa g_t^\top + A_{t+1}^\top E_{t+1}, \quad t < L. \tag{30}$$

For one mode,

$$\mathrm{d}H_t = A_t\,\mathrm{d}H_{t-1} + (\mathrm{d}A_t)H_{t-1} + e_1 v_t^\top\,\mathrm{d}\eta_t^{\mathrm{w}} + \eta_t^{\mathrm{w}} e_1(\mathrm{d}v_t)^\top.$$

The transition differential is

$$\mathrm{d}A_t = \frac{A_t}{\rho}\,\mathrm{d}\rho - A_t\,\mathrm{d}\Delta_t + \rho \mathrm{e}^{-\Delta_t} R'(\theta)\,\mathrm{d}\theta. \tag{31}$$

Using

$$\langle E_t, A_t\,\mathrm{d}H_{t-1}\rangle_F = \langle A_t^\top E_t, \mathrm{d}H_{t-1}\rangle_F,$$

the core derivatives are

$$\begin{aligned}
\nabla_{v_t}\mathcal{L} &= \sum_i \eta_{i,t}^{\mathrm{w}} E_{i,t}^\top e_1, & \partial_{\eta_t^{\mathrm{w}}}\mathcal{L} &= e_1^\top E_t v_t,\\
\partial_{\eta_t^{\mathrm{r}}}\mathcal{L} &= g_t^\top H_t^\top \kappa, & \partial_{\Delta_t}\mathcal{L} &= -\langle E_t, A_t H_{t-1}\rangle_F,\\
\partial_\rho \mathcal{L} &= \sum_t \langle E_t, A_t H_{t-1}\rangle_F/\rho, & \partial_\theta \mathcal{L} &= \sum_t \rho \mathrm{e}^{-\Delta_t}\langle E_t, R'(\theta) H_{t-1}\rangle_F,\\
\nabla_\kappa \mathcal{L} &= \sum_t \eta_t^{\mathrm{r}} H_t g_t.
\end{aligned} \tag{32}$$

Here

$$\langle X, Y\rangle_F = \mathrm{tr}(X^\top Y).$$

Because

$$\kappa = (\kappa^{\mathrm{c}}, -\kappa^{\mathrm{s}})^\top,$$

the stored readout coefficients satisfy

$$\frac{\partial \mathcal{L}}{\partial \kappa^{\mathrm{c}}} = [\nabla_\kappa \mathcal{L}]_1, \qquad \frac{\partial \mathcal{L}}{\partial \kappa^{\mathrm{s}}} = -[\nabla_\kappa \mathcal{L}]_2,$$

before adding the zero-lag correction to the κ^{c} derivative.

The zero-lag correction is local and does not change the state adjoint. For one mode, define

$$\chi_t = g_t^\top v_t, \qquad \ell_t^{\mathrm{corr}} = -a\eta_t^{\mathrm{r}}\eta_t^{\mathrm{w}}\kappa^{\mathrm{c}}\chi_t.$$

Its nonzero derivatives are

$$\begin{aligned}
\nabla_{v_t}\ell_t^{\mathrm{corr}} &= -a\eta_t^{\mathrm{r}}\eta_t^{\mathrm{w}}\kappa^{\mathrm{c}} g_t, & \partial_a \ell_t^{\mathrm{corr}} &= -\eta_t^{\mathrm{r}}\eta_t^{\mathrm{w}}\kappa^{\mathrm{c}}\chi_t,\\
\partial_{\eta_t^{\mathrm{w}}}\ell_t^{\mathrm{corr}} &= -a\eta_t^{\mathrm{r}}\kappa^{\mathrm{c}}\chi_t, & \partial_{\eta_t^{\mathrm{r}}}\ell_t^{\mathrm{corr}} &= -a\eta_t^{\mathrm{w}}\kappa^{\mathrm{c}}\chi_t,\\
\partial_{\kappa^{\mathrm{c}}}\ell_t^{\mathrm{corr}} &= -a\eta_t^{\mathrm{r}}\eta_t^{\mathrm{w}}\chi_t.
\end{aligned}$$

The correction has no direct derivative with respect to Δ, ρ, θ, or κ^{s}.

For the raw decay parameter,

$$\rho = \exp[-\mathrm{softplus}(\tilde{\rho})], \qquad \frac{\partial \rho}{\partial \tilde{\rho}} = -\rho\,\mathrm{sigmoid}(\tilde{\rho}).$$

For the affine logits $s_t^\Delta, s_t^{\mathrm{w}}, s_t^{\mathrm{r}}$ from Appendix C.2,

$$\begin{aligned}
\partial_{\tilde{\rho}}\mathcal{L} &= -\,\mathrm{sigmoid}(\tilde{\rho})\sum_t \langle E_t, A_t H_{t-1}\rangle_F,\\
\partial_{s_t^\Delta}\mathcal{L} &= \mathrm{sigmoid}(s_t^\Delta)\partial_{\Delta_t}\mathcal{L},\\
\partial_{s_t^{\mathrm{w}}}\mathcal{L} &= \eta_t^{\mathrm{w}}(1-\eta_t^{\mathrm{w}}/2)\partial_{\eta_t^{\mathrm{w}}}\mathcal{L},\\
\partial_{s_t^{\mathrm{r}}}\mathcal{L} &= \eta_t^{\mathrm{r}}(1-\eta_t^{\mathrm{r}}/2)\partial_{\eta_t^{\mathrm{r}}}\mathcal{L}.
\end{aligned} \tag{33}$$

The gate derivatives in Equation 33 include the local zero-lag terms above.

Stable factorization of accumulated decay. For one mode,

$$-\log \|A_t \cdots A_{s+1}\|_2 = -(t-s)\log\rho + q(t) - q(s). \tag{34}$$

For positions t in a chunk beginning at b, define the zero-based within-chunk coordinate $t-b$ and

$$\ell(t) = (t-b)\gamma + \sum_{u=b}^{t} \Delta(u).$$

For any center $\ell_\star$,

$$\rho^{t-s}\mathrm{e}^{-(q(t)-q(s))} = \mathrm{e}^{-(\ell(t)-\ell_\star)}\mathrm{e}^{\ell(s)-\ell_\star}.$$

The rotation factor separates similarly:

$$R(\theta(t-s)) = R(\theta(t-b))R(\theta(s-b))^\top.$$

These identities allow endpoint-factorized evaluation without changing the mode response.

Centering reduces the dynamic range of the endpoint factors. The factorized branch is used only when its centered factors remain inside the configured fp32 range; otherwise the kernel evaluates the causal within-chunk interaction sums directly using a masked $Q \times Q$ block. With largest and smallest positive normal fp32 values $F_{\max}$ and $F_{\min}$, define

$$E_{\mathrm{safe}} = \min\{\log(F_{\max}\epsilon), -\log(F_{\min}/\epsilon)\}, \qquad \epsilon = 2 \times 10^{-3}.$$

The factorized branch admits a mode when

$$(Q-1)\gamma_{k,i} + \max_j \sum_{t\in\mathcal{C}_j} \Delta_{k,i}(t) \le 2E_{\mathrm{safe}}.$$

Intrinsic decay is centered at $(Q-1)/2$, and each chunk's accumulated clock is centered at half its final sum. The maximum is taken across chunks and batch sequences, giving one admission decision per channel and mode. Modes outside this range use the direct branch.

Algorithm 3 organizes the reverse pass through the chunk boundaries. Its direct branch is written as a reverse recurrence to specify the gradients; the kernel evaluates the equivalent masked within-chunk sums. Let $\Lambda_{k,i,j}$ denote the gradient from chunks after j with respect to the mode state at e_j. Each chunk defines

$$\mathcal{P}_{k,i,j} = \Pi_{k,i}(e_j, b_j - 1)$$

and a local contribution

$$\mathcal{B}_{k,i,j} = \sum_{t=b_j}^{e_j} \eta^{\mathrm{r}}_{k,i}(t)\Pi_{k,i}(t, b_j-1)^\top \kappa_{k,i} g_k(t)^\top.$$

The boundary adjoints then satisfy

$$\Lambda_{k,i,j-1} = \mathcal{B}_{k,i,j} + \mathcal{P}^\top_{k,i,j}\Lambda_{k,i,j}.$$

Algorithm 3: Backward evaluation with boundary adjoints

Input: $g_k(t) = \partial\mathcal{L}/\partial z_k(t)$, forward inputs and parameters, $\mathcal{H}$, chunks $\mathcal{C}_j = [b_j, e_j]$, $j = 1, \dots, J$
Output: gradients of values, gate/clock logits, and modal parameters

foreach (k, i, j) *in parallel* **do**
  $\mathcal{P}_{k,i,j} \leftarrow \Pi_{k,i}(e_j, b_j - 1)$
  $\mathcal{B}_{k,i,j} \leftarrow \sum_{t=b_j}^{e_j} \eta^{\mathrm{r}}_{k,i}(t)\Pi_{k,i}(t, b_j - 1)^\top \kappa_{k,i} g_k(t)^\top$
$\Lambda_{k,i,J} \leftarrow 0$ for all (k, i)
for $j = J, J-1, \dots, 1$ **do**
  $\Lambda_{k,i,j-1} \leftarrow \mathcal{B}_{k,i,j} + \mathcal{P}^\top_{k,i,j}\Lambda_{k,i,j}$ for all (k, i)
foreach (k, i, j) *in parallel* **do**
  $H_{k,i}(b_j - 1{:}e_j) \leftarrow \mathrm{Recompute}(\mathcal{H}_{k,i,j}, v, \eta^{\mathrm{w}}, A)$
  if $\mathrm{Admit}(\rho_{k,i}, \{\Delta_{k,i}[\mathcal{C}_c]\}_{c=1}^{J}, \mathrm{dtype})$ **then**
    $\mathcal{D}_{k,i,j} \leftarrow \mathrm{FactorizedGrad}(H, \Lambda_{k,i,j}, g; \mathcal{C}_j)$
  else
    $E \leftarrow \Lambda_{k,i,j}$
    $\mathcal{D}_{k,i,j} \leftarrow 0$
    for $t = e_j, e_j - 1, \dots, b_j$ **do**
      $E \leftarrow E + \eta^{\mathrm{r}}_{k,i}(t)\kappa_{k,i} g_k(t)^\top$
      $\mathcal{D}_{k,i,j} \leftarrow \mathcal{D}_{k,i,j} + \mathrm{CoreGrad}(t, H_t, H_{t-1}, E; g, \eta^{\mathrm{w}}, \eta^{\mathrm{r}}, A, \kappa)$
      $E \leftarrow A_{k,i}(t)^\top E$
  $\mathcal{D}_{k,i,j} \leftarrow \mathcal{D}_{k,i,j} + \mathrm{CorrectionGrad}(g, v, \eta^{\mathrm{w}}, \eta^{\mathrm{r}}, a, \kappa^{\mathrm{c}}; \mathcal{C}_j)$
$\nabla\mathcal{L} \leftarrow \mathrm{ReduceShared}(\{\mathcal{D}_{k,i,j}\})$
$\nabla\mathcal{L} \leftarrow \mathrm{ChainRules}(\nabla\mathcal{L}; \mathrm{sigmoid}, \mathrm{softplus}, \rho, \kappa)$
return $\nabla\mathcal{L}$

O.3 Fused single-token decoder

The tokenwise decoder uses the same recurrence as the chunked parallel scan, together with the channel-local value blend and zero-lag correction. Values and the write, read, and clock logits are produced by one concatenated projection, followed by the recurrent update and output projection. The identity

$$\exp[-\mathrm{softplus}(x)] = \mathrm{sigmoid}(-x)$$

gives the clock contribution to the transition directly.

The rotation matrices $R(\theta_{k,i})$ and other fixed modal constants are cached when a tier is loaded. The timed fused decoder includes the trained value blend and zero-lag correction.

Algorithm 4 receives the recurrent states and window KV caches for positions $1, \dots, t-1$ and processes token t. Project computes the retained channel values and gate/clock logits, Blend is Equation 23, and Attn denotes causal multi-head attention over the retained window.

Algorithm 4: One decoding step of a standalone export

Input: token id_t, Θ_T, states H, window KV caches $(\mathcal{K}, \mathcal{V})$, attention-layer set $\mathcal{A}$, window w, position t
Output: next-token logits and updated $H, \mathcal{K}, \mathcal{V}, t$

$x \leftarrow E_{\mathrm{tok}}[\mathrm{id}_t, :]^\top$
for $\ell = 1, \ldots, N_{\mathrm{layers}}$ **do**
  $u \leftarrow \mathrm{RMSNorm}_{\ell,1}(x)$
  if $\ell \notin \mathcal{A}$ **then**
    $(v^0, s^{\mathrm{w}}, s^{\mathrm{r}}, s^{\Delta}) \leftarrow \mathrm{Project}_{\ell,T}(u)$
    for $k = 1, \ldots, K_T$ **do**
      $v_k \leftarrow \mathrm{Blend}_{\beta_k}(v_k^0)$
      $z_k \leftarrow 0$
      for $i = 1, \ldots, m$ **do**
        $(\eta^{\mathrm{w}}, \eta^{\mathrm{r}}) \leftarrow (2\,\mathrm{sigmoid}(s^{\mathrm{w}}_{k,i}), 2\,\mathrm{sigmoid}(s^{\mathrm{r}}_{k,i}))$
        $\Delta \leftarrow \mathrm{softplus}(s^{\Delta}_{k,i})$
        $A \leftarrow \rho_{k,i} \mathrm{e}^{-\Delta} R(\theta_{k,i})$
        $H_{\ell,k,i} \leftarrow A H_{\ell,k,i} + \eta^{\mathrm{w}} e_1 v_k^\top$
        $z_k \leftarrow z_k + \eta^{\mathrm{r}} \left(H_{\ell,k,i}^\top \kappa_{k,i} - a_{k,i} \eta^{\mathrm{w}} \kappa^{\mathrm{c}}_{k,i} v_k \right)$
    $\hat{y} \leftarrow K_T^{-1/2} \sum_k W_k^{\mathrm{out}} z_k$
  else
    $(\mathbf{q}, \mathbf{k}, \mathbf{v}) \leftarrow \mathrm{QKV}_\ell(u)$
    $(\mathbf{q}, \mathbf{k}) \leftarrow \mathrm{RoPE}_t(\mathbf{q}, \mathbf{k})$
    $(\mathcal{K}_\ell, \mathcal{V}_\ell) \leftarrow \mathrm{Append}(\mathcal{K}_\ell, \mathcal{V}_\ell; \mathbf{k}, \mathbf{v})$
    if $|\mathcal{K}_\ell| > w$ **then**
      $(\mathcal{K}_\ell, \mathcal{V}_\ell) \leftarrow (\mathcal{K}_\ell[-w:], \mathcal{V}_\ell[-w:])$
    $\hat{y} \leftarrow \mathrm{OutProj}_\ell\left(\mathrm{Attn}_\ell(\mathbf{q}, \mathcal{K}_\ell, \mathcal{V}_\ell)\right)$
  $x \leftarrow x + \hat{y}$
  $u \leftarrow \mathrm{RMSNorm}_{\ell,2}(x)$
  $n \leftarrow n_{\mathrm{ff},T}$
  $z^{\mathrm{ff}} \leftarrow \mathrm{SiLU}(W^{\mathrm{gate}}_{\ell,1:n,:} u) \odot (W^{\mathrm{up}}_{\ell,1:n,:} u)$
  $x \leftarrow x + W^{\mathrm{down}}_{\ell,:,1:n} z^{\mathrm{ff}}$
logits $\leftarrow E_{\mathrm{tok}}\, \mathrm{RMSNorm}_{\mathrm{final}}(x)$
$t \leftarrow t + 1$
return logits, $H, \mathcal{K}, \mathcal{V}, t$

O.4 Device-adaptive autotuning

Kernel configurations are selected separately for each device and shape class. Candidate warp counts, pipeline stages and band-kernel variants are timed at the fixed chunk size under the same workload and precision, and the selected configuration is checked against the reference implementation for forward and gradient agreement.

O.5 Numerical correctness checks

Independent fp64 CPU implementations compare direct recurrence, explicit interaction matrices, and chunked execution under varying clocks, signed sine coefficients, partial chunks, nonzero entering states, and zero-lag corrections. Analytic gradients are compared with centered finite differences. Separate checks verify standalone slicing against the same-tier shared model and compare the fused GPU decoder with both the chunked forward path and a reference tokenwise stepper.

P Systems Evaluation

The systems study reports whole-model training, prefill, decoding, and memory. Shape benchmarks measure the executable architectures at the stated parameter counts, precision, and execution settings. Table 1(b) places trained-checkpoint quality beside the corresponding tier-size benchmarks. The A100 study is complemented by H200 and B300 inference measurements. The single-GPU H200 sweeps are reported separately from the multi-GPU production-training logs.

Table 49: Numerical verification of the recurrent and export implementations. CPU entries report fp64 maximum absolute error with random seed 2333; centered finite differences use step 10^{-6}. GPU entries report relative error on the recorded reduced test shapes.

Check	Error metric	Maximum
CPU recurrence, chunk boundaries and zero-lag identities	absolute	3.55×10^{-15}
CPU recurrence gradients versus centered finite differences	absolute	2.28×10^{-9}
CPU toy export versus same-tier shared model	absolute	0
GPU fused decoder versus chunked forward	relative	1.37×10^{-6}
GPU fused decoder versus reference stepper	relative	2.11×10^{-5}

P.1 Training throughput per device

Table 50 reports full-capacity throughput at context 2048, using steps 20–35 of 40-step probes and the listed micro-batch and accumulation settings. At 370M, ESSH reaches 41.7 versus 34.1 ktok/s for Mamba-3 + SWA on A100, while the H200 ordering is reversed at 75.5 versus 88.6 ktok/s. The comparison describes device-specific training throughput.

The family-cost measurements in Appendices N.7 and N.8 include capacity mixing and teacher computation. Their pretraining costs are estimated from the rate under that workload, while shared refinement uses measured elapsed time.

Table 50: Full-capacity training throughput on one GPU at context 2048. Params include tied embeddings. B/A denote micro-batch size/accumulation; rates use steps 20–35 of 40-step probes. Only the 370M group provides near-size cross-model comparisons. The 1.53B block reports ESSH device throughput alone. ESSH remains in full-capacity warm-up, before capacity mixing and teacher forwards.

Device	Model	Params (M)	B	A	ktok/s ↑
370M-scale full models					
A100	ESSH 370M	370.9	8	16	**41.7**
A100	Mamba-3 + SWA 377M	377.1	8	16	34.1
H200	ESSH 370M	370.9	8	16	75.5
H200	Mamba-3 + SWA 377M	377.1	8	16	**88.6**
B300	ESSH 370M	370.9	8	16	82.3
1.5B-scale full models					
A100	ESSH 1.53B	1531.1	4	8	13.5
H200	ESSH 1.53B	1531.1	4	8	22.4
B300	ESSH 1.53B	1531.1	4	8	27.2

P.2 Prefill

The A100 prefill curves separate the cost of adding tokens from the cost of increasing model capacity. Within each tier, throughput rises slightly as context grows from 2K to 128K; full ESSH increases from 28.3 to 34.0 ktok/s. Across tiers, smaller retained shapes are faster. The recurrence uses fixed-size mode states, and attention uses a fixed window. Their work grows linearly with sequence length.

The advantage of a compact tier therefore persists for long inputs: it removes spectral-channel and FFN work at every position. Peak allocation also contains the full vocabulary-logit tensor returned by this benchmark, so it follows a different curve from throughput and recurrent state. The 2K latency measures the prefill component of time to first token. At matched full-model size, Mamba-2 has the highest measured prefill throughput. ESSH overtakes Transformer++ as the input grows: their 128K rates are 34.0 and 11.0 ktok/s, respectively. This crossing follows the different growth of recurrent/windowed processing and full attention; it is distinct from the recurrent decode comparison.

Table 51: A100 batch-one full-logit prefill. Throughput is ktok/s and latency ms; peak allocations are reported in Table 53. Compact ESSH rows use the tier sweep; full-model rows use matched three-session measurements. Each session uses three warm-ups and five timed calls. Configurations are in Table 54; Mamba-3 MIMO uses the compiler-compatible repeat. Bold compares measured entries within each ruled group.

		prefill ktok/s ↑				
Model	Params (B)	2K	8K	32K	128K	Prefill ms (2K) ↓
ESSH, T_1	0.474	**44.4**	**46.7**	**48.5**	**49.1**	**46.1**
ESSH, T_4	0.828	36.4	39.3	41.1	41.6	56.3
ESSH, T_7	1.168	32.1	35.1	36.2	36.5	63.8
ESSH, T_{10}	1.531	28.3	32.0	33.5	34.0	72.3
Mamba-2 + FFN	1.526	**47.5**	**53.6**	**55.0**	**54.7**	**43.1**
Mamba-3 SISO + FFN	1.535	32.5	36.3	37.5	37.7	63.1
Mamba-3 MIMO + FFN	1.527	26.7	29.7	30.4	30.5	76.6
Transformer++	1.536	39.0	37.2	26.0	11.0	52.5

P.3 Decode latency, throughput and memory

Decode latency is the time for one synchronous step across the entire batch. Aggregate throughput is the number of sequences divided by that step time. The matched A100 comparison gives ESSH the lowest measured latency at all four batch sizes: 3.06 ms at batch one and 31.93 ms at batch 512. Step time grows much more slowly than batch size, allowing the same model weights and operations to serve more generated tokens per invocation. At batch one, the tested Mamba implementations take 1.45–1.78 times ESSH's latency, and Transformer++ takes 1.92 times as long.

Within ESSH, T_1 roughly halves batch-one latency relative to T_{10}, but the benefit is smaller at batch 128. Truncation removes spectral-channel and FFN computation; the layer sequence, residual width, embeddings, and window attention remain. These retained operations set a common cost across tiers, so the 69% parameter reduction produces a smaller latency reduction. The measured family is consequently useful for choosing both model size and response time.

The memory decomposition makes the same resource allocation explicit. Recurrent state shrinks from 3.28 to 0.82 MB, but the window KV cache remains 50.33 MB. Most model-storage savings from truncation therefore come from weights, while the fixed cache reserves direct context access at every tier. The fixed window KV cache supplies the same local access route as spectral-channel count and feed-forward width decrease.

P.4 Where the decode advantage comes from

The fused-versus-unfused A100 control isolates the benefit of the fused implementation. Both paths execute the 1.531B ESSH model, yet batch-one graph latency decreases from 7.40 to 3.02 ms. The mathematical predictor and parameter count are fixed. The structured recurrence permits these projections, channel-local updates, and readouts to be combined in the fused recurrent path.

The B300 profile shows how the deployed computation differs from a near-size external model. ESSH and Mamba-2 contain 1.531B and 1.526B parameters, but their recurrent steps execute 367 and 963 compute kernels, respectively. Counts are GPU compute-kernel events in a 20-step profile of the eager/compiled step function, divided by 20; graph latency is timed separately. Shared channel values are reused across mode updates, and the structured transitions allow the resulting operations to be fused. The parameter-matched timing and the fused-versus-unfused control together support efficient execution as a contribution of the implemented recurrence.

Table 54 records the comparison configurations: depth, residual width, vocabulary, and embedding tying are held fixed, and baseline FFN widths are selected before timing to match total parameters within 0.34%. These architecture timings use randomly initialized weights at the stated tensor shapes; quality is evaluated from trained models. Table 61 gives the separate near-size comparison at 370M.

Table 52: A100 whole-model decode latency (ms/step), with state/cache sized for a 2K context. Batch columns use CUDA graphs; the last uses eager execution. ESSH weights/KV are bf16 and recurrent state fp32. Full-model comparisons use matched three-session measurements, with MIMO from the compiler-compatible repeat. Compact tiers and the fused/unfused control use the separate protocols in Appendix E.5. Configurations: Table 54. Compact and full-model blocks are ranked separately; OOM denotes out of memory.

Model	Params (B)	$B = 1$	16	128	512	Eager $B = 1$
ESSH, T_1	0.474	**1.46**	**2.60**	7.53	**23.76**	**3.98**
ESSH, T_2	0.575	1.67	2.66	**7.35**	24.89	4.07
ESSH, T_3	0.716	1.95	2.96	7.75	25.91	4.20
ESSH, T_4	0.828	2.12	3.02	8.29	27.31	4.27
ESSH, T_5	0.931	2.22	3.13	8.40	27.73	4.25
ESSH, T_6	1.076	2.41	3.28	8.66	28.82	4.28
ESSH, T_7	1.168	2.62	3.44	8.88	30.51	4.43
ESSH, T_8	1.291	2.75	3.63	9.36	31.08	4.40
ESSH, T_9	1.416	2.97	3.76	9.43	31.70	4.44
ESSH, T_{10}	1.531	**3.06**	**3.86**	**9.44**	**31.93**	**4.47**
Mamba-2 + FFN	1.526	4.44	6.08	12.49	38.63	21.39
Mamba-3 SISO + FFN	1.535	5.09	7.21	16.70	53.26	31.51
Mamba-3 MIMO + FFN	1.527	5.45	7.86	18.01	56.19	33.44
Transformer++	1.536	5.87	12.47	46.83	OOM	17.96
Fused-versus-unfused control						
ESSH fused	1.531	**3.02**	**3.92**	**9.91**	**32.07**	**4.53**
ESSH unfused	1.531	7.40	9.24	16.80	47.09	27.28

Table 53: A100 memory at the stated model sizes. State/cache totals exclude weights; prefill peaks include weights, logits, and temporaries. ESSH state is fp32 and cache bf16. MB/GB are decimal. Full-model allocations use the matched A100 measurements; compact rows use the tier sweep. Bold compares measured total/peak allocations separately within compact and full-model blocks.

		n_b=1 (MB) ↓			total n_b=128 ↓	prefill peak (GB) ↓			
Model	Params (B)	state	KV	total	(GB)	2K	8K	32K	128K
ESSH, T_1	0.474	0.82	50.33	**51.15**	**6.55**	**1.5**	**3.1**	**9.6**	**35.7**
ESSH, T_4	0.828	1.84	50.33	52.17	6.68	2.2	3.8	10.3	36.4
ESSH, T_7	1.168	2.56	50.33	52.89	6.77	3.0	4.6	11.1	37.1
ESSH, T_{10}	1.531	3.28	50.33	53.61	6.86	**3.6**	**5.2**	**11.7**	**37.8**
Mamba-2 + FFN	1.526	30.33	0.00	**30.33**	**3.88**	**3.6**	5.3	11.8	**37.8**
Mamba-3 SISO + FFN	1.535	59.64	0.00	59.64	7.63	**3.6**	5.3	11.8	**37.8**
Mamba-3 MIMO + FFN	1.527	61.01	0.00	61.01	7.81	**3.6**	**5.2**	**11.7**	**37.8**
Transformer++	1.536	0.00	469.99	469.99	60.16	**3.6**	**5.2**	11.8	**37.8**

Table 54: Configurations for the A100, H200, and B300 comparisons. Parameter counts include tied embeddings. All use vocabulary 128256, residual width 2048, and 28 blocks. Baseline FFN widths are matched in multiples of 64; official Mamba mixer settings stay fixed. Gap is relative to ESSH. A100 Mamba-3 MIMO uses the compiler-compatible repeat.

Model	Params (M)	Gap (%)	FFN width	SWA layers
ESSH	1531.1	+0.00	5632	3
Mamba-2 + FFN	1526.0	-0.33	3136	0
Mamba-3 SISO + FFN	1534.9	+0.25	3136	0
Mamba-3 MIMO + FFN	1526.7	-0.29	2816	0
Transformer++	1536.3	+0.34	4672	0

P.5 Cross-device comparison

Appendix R reports A100, H200, and B300 measurements, with the consolidated efficiency table at the end of the appendix. The single-GPU H200 comparisons are separate from the eight-GPU production-training logs.

Q Reproducibility

This section collects the configurations, export procedure, evaluation environment, and compute accounting needed to rerun the experiments. The supplementary material contains the experimental code.

Q.1 Configurations and hyperparameters

Table 55 collects the recipe constants. The launch configurations are in Appendix E.2. Ablation tables state deviations. The reported 1.5B run uses the constants chosen before it was launched. Later 370M ablations do not change that configuration.

Table 55: Constants used in the main recipe and the experiment sections that evaluate alternative settings.

Constant	370M	1.5B	fixed by
$\bar{K}, m$	32, 8	32, 8	divisibility, Appendices L.5, L.4
window layers, window	3 late, 2048	3 late, 2048	Appendix E.2
channel / feed-forward exponents	0.5 / 0.8	same	Appendix M.1
full share f, slice scale α_{ff}	0.75, 0.5	0.75, 0.5	Appendix M.2
distillation objectives and coefficients	KL at 0.5 (pretraining), TV at 5 (final stage)	same	Appendix M.2
pretraining capacities	$1/16$, $3/32$, $1/8$, $5/32$, $3/16$, $1/4$, $3/8$, $1/2$, $3/4$, 1	same	design
ordering period, window	500 steps, until 80%	same	Appendix D.3
clock bias, gate init	-3, gates at one	same	Appendix C.2
zero-lag coefficient initialization	0.5, trainable	same	Appendix C.3
peak lr, warm-up, floor	1.2×10^{-3}, 1%, 10%	1.0×10^{-3}, 1%, 10%	Appendix M.2
AdamW betas, wd, clip	(0.9, 0.95), 0.1, 1.0	same	Mamba-2 recipe
tokens per step, context	262,144, 2048	524,288, 2048	design
final stage	TV 5, 2060 steps, 131,072 tokens/step (0.27B), lr 6×10^{-5}	TV 5, 4120 steps, same tokens/step (0.54B), same lr	training configuration
SR capacities	$\{2, 4, 8, 16, 24\}/32$, one per step	T_1–T_9, shuffled across micro-batches	Appendix D.8

The shared refinement objective is defined in Appendix D.8. Unlike the full-tier distillation step, this stage includes the full model's label loss as well as the reduced-capacity model's label and distillation losses.

Q.2 Export procedure

Given a trained full model, instantiate each tier using Table 6. Slice its tensors as described in Appendix D.2 and load them into the standalone model with strict key matching. Verify the parameter count and compare its outputs with those of the shared model at the same tier.

Q.3 Evaluation harness and environment

The reproducibility record is experiment specific. It includes framework, compiler, driver, attention and recurrent-kernel revisions, evaluation-harness version, tokenizer revision, task configuration, prompt template, and data fingerprint (Paszke et al., 2019; Tillet et al., 2019). The seven-task metric selection is defined in Appendix E.4; evaluator versions are retained in the corresponding run records. Framework and CUDA versions are recorded per run. The matched A100 measurements use PyTorch 2.11.0+cu128 and CUDA 12.8. The matched B300 cohort uses PyTorch 2.11.0+cu130 and CUDA 13.0; the H200 cohort uses PyTorch 2.11.0+cu128 and CUDA 12.8. Both use Triton 3.6.0. The 48 B300 jobs and 24 H200 jobs each share one device and a consistent set of source revisions within their cohort.

Every reported result is identified by the training run, training seed, checkpoint stage, deployment tier, and metric field. The final 1.5B ESSH model, 370M comparison models, and systems configurations have separate source records. The result ledger retains the source path and reported value. The final 1.5B model is identified by its SHA-256 hash, and each reported score is linked to its tier and source file.

Training-seed ranges and item-cluster bootstrap intervals describe different uncertainty sources. The former compare independently trained models. The latter resample the key pairs within a fixed needle evaluation. Neither substitutes for the other.

Q.4 Compute accounting

Table 56 reconstructs the compute used for this study from scheduler accounting and run logs, including their failed, aborted, and repeated segments. Including the weighted-CE control, the total is 4508.6 raw GPU-hours, displayed as 4509 after rounding. It sums hours across device types without converting them to A100-equivalent compute and excludes experiments outside the reported study.

The 1.5B main run uses H100 and A100 segments and two eight-H200 segments. Its cost also includes short aborted H800 and B200 launches. The 7.4B shared-recipe row covers the reported seeds 197 and 511. Systems costs cover timing runs and their recorded telemetry; there is no separate energy benchmark. Individual records identify the device, job or log source, and counted duration, including reconstructed intervals where complete timestamps were unavailable.

Table 56: Reconstructed raw GPU-hours for this study. Totals include failed, aborted, and repeated work for these experiments (105.2 GPU-hours), while excluding unreported exploration. Device hours are summed without hardware normalization and rounded to integers.

Stage	Devices	GPU-hours
1.5B main training	A100, H100, H200, H800, B200	1808
1.5B final stage, export and evaluation	H200, H800, RTX 5090/3090	47
370M shared recipe, 7.4B	A100, H100, RTX 5090/3090	104
370M comparisons, 7.4B	A100, H200, RTX 5090/3090	772
370M ablations, 3.7B	A100, RTX 5090/3090	1633
Synthetic tasks and DNA	A100, RTX 5090/3090	129
Systems measurements	A100, H200, B300	16
Total		4509

R Cross-Device Efficiency

The systems comparisons use 28-layer, width-2048 configurations on A100-SXM4-80GB, NVIDIA H200, and NVIDIA B300. The timed ESSH configuration places window attention at blocks 9, 19, and 27; the final quality model uses blocks 15, 22, and 27 (Table 7). Both have the stated parameter and state+KV counts. The measurements characterize the explicitly specified timing configuration.

Decode uses CUDA graphs after a 2048-token context, with 50 warm-ups and 200 timed steps per session. Matched full-model results on all three devices and B300 compact batch-one results are

medians of three sessions. Prefill uses three warm-ups and five timed calls per session. Tables identify single-session compact measurements where retained. The matched H200/B300 runs use the same attention-backend policy for all models, with cuDNN SDPA disabled.

A100 Mamba-3 MIMO uses a separate three-session run compiled with GCC 12.4; its paired ESSH graph latencies differ by at most 0.42% from the main A100 run. Ratios in the text use unrounded measurements.

R.1 H200 SINGLE-GPU MEASUREMENTS

The parameter-matched H200 comparison repeats the decode advantage on a second GPU generation. At batch one, ESSH takes 1.67 ms, versus 2.91 for Mamba-2, 3.52 for Mamba-3 SISO, 3.85 for Mamba-3 MIMO, and 4.15 for Transformer++. The gain is 1.75–$2.31\times$ over the tested Mamba implementations and $2.49\times$ over Transformer++. ESSH remains the fastest measured full model at every batch size. Together with the B300 results, this shows that the fused recurrent implementation is effective at the same parameter budget on both devices.

The prefill ordering changes with input length. Transformer++ is faster at 2K and 8K, while ESSH sustains 60.7 ktok/s at 128K versus 18.4, a $3.31\times$ advantage. The chunked parallel scan and fixed-window attention require linear work as context length grows, consistent with this sustained throughput. Mamba-2 reaches 125.4 ktok/s at 128K, retaining its strength in parallel prefill, while ESSH has the lowest recurrent decode latency. These measurements distinguish the execution benefits of the two workloads.

Cache size becomes a feasibility constraint at large batches. Transformer++ encounters allocation failure at batch 512 in all three sessions: its 2K KV cache would require 240.6 GB, exceeding the 150.0 GB (139.7 GiB) reported by the driver. ESSH completes that setting at 17.63 ms with 27.45 GB of state+KV. Keeping attention in three window layers therefore reserves a substantially smaller per-sequence cache while the remaining layers use recurrent state. Capacity-dependent timing is reported for A100 and B300 under the protocols specified in their tables.

Table 57: H200 decode latency (ms/step), with a 2K cache. Batch columns use CUDA graphs; the last uses eager execution. All five models use the parameter-matched three-session measurements and the same attention-backend policy. ESSH state is fp32 and weights/KV bf16; baselines retain native state precision. OOM records allocation failure in all three sessions. Bold ranks within each group.

Model	Params (B)	$B = 1$	16	128	512	Eager $B = 1$
Parameter-matched full models						
ESSH, T_{10}	1.531	**1.67**	**2.15**	**5.35**	**17.63**	**3.44**
Mamba-2 + FFN	1.526	2.91	3.93	7.81	22.05	15.97
Mamba-3 SISO + FFN	1.535	3.52	4.78	8.85	24.12	27.65
Mamba-3 MIMO + FFN	1.527	3.85	5.30	9.87	26.46	28.71
Transformer++	1.536	4.15	7.32	30.03	OOM	12.54

Table 58: H200 batch-one full-logit prefill. Throughput: ktok/s; latency: ms; peak allocation: decimal GB. All five models use three sessions with three warm-ups and five timed calls per session. cuDNN SDPA is disabled for the shared comparison. Bold ranks separately within each group.

		Prefill ktok/s ↑				ms ↓	Peak GB ↓	
Model	Params (B)	2K	8K	32K	128K	2K	2K	128K
Parameter-matched full models								
ESSH, T_{10}	1.531	54.9	58.9	60.6	60.7	37.3	**3.64**	**37.79**
Mamba-2 + FFN	1.526	73.8	**126.0**	**127.4**	**125.4**	27.8	3.65	37.80
Mamba-3 SISO + FFN	1.535	71.8	81.1	83.8	83.9	28.5	3.65	37.80
Mamba-3 MIMO + FFN	1.527	39.6	42.1	42.6	42.2	51.7	**3.64**	**37.79**
Transformer++	1.536	**89.0**	81.5	50.0	18.4	**23.0**	3.65	37.80

R.2 B300 DECODING ACROSS THE TEN TIERS

At near-equal parameter counts, ESSH has the lowest decode latency at every measured batch size (Table 59). Batch-one latency is 1.37 ms, compared with 2.93 for Mamba-2, 3.60 for Mamba-3 SISO, 3.84 for Mamba-3 MIMO, and 4.15 for Transformer++. All use the same depth, residual width, vocabulary, and timing code, and parameter counts differ from ESSH by at most 0.34%.

The gain is present even though ESSH stores slightly more weights than Mamba-2, 3.062 versus 3.052 GB. Their compute-kernel counts differ much more, 367 versus 963 per step. The spectral computation shares each projected channel value across its modes and combines state updates and readouts in the fused recurrent path. Together with the fused-versus-unfused control, the profile links the measured gain to an execution design that carries out the recurrent predictor with fewer compute-kernel launches.

The relative advantage narrows with batch size: ESSH is $2.14\times$ faster than Mamba-2 at batch one and reaches 42.0 versus 28.4 ktok/s for the fastest baseline, Mamba-3 SISO, at batch 512. Batching shares fixed model work across more generated tokens, so the comparison moves from short-response latency toward sustained aggregate throughput. ESSH remains favorable in both regimes.

Within the ESSH family, removing about 69% of the parameters changes latency from 1.37 to 1.13 ms at T_1. The unchanged backbone and attention operations remain in every tier, and batch-one latency is nearly flat over T_1–T_3 (1.12–1.13 ms). Parameter savings therefore exceed the measured latency savings in this range.

Table 59: B300 decode latency (ms/step), after a 2K context. Batch columns use CUDA graphs; the last uses eager execution. All full-model entries and compact $B = 1$ entries use three-session measurements. Compact $B = 16/128/512$ entries use a separate tier sweep. ESSH state is fp32; weights/KV are bf16. Mamba rows use the official mixers with a residual FFN block. Bold is ranked separately within each ruled group.

Model	Params (B)	$B = 1$	16	128	512	Eager $B = 1$
ESSH, T_1	0.474	1.13	1.35	**3.30**	**10.25**	3.43
ESSH, T_2	0.575	**1.12**	**1.33**	3.53	10.62	**3.41**
ESSH, T_3	0.716	1.13	1.36	3.55	10.91	3.46
ESSH, T_4	0.828	1.20	1.42	3.63	11.18	3.49
ESSH, T_5	0.931	1.18	1.42	3.61	11.13	3.49
ESSH, T_6	1.076	1.28	1.49	3.83	11.62	3.46
ESSH, T_7	1.168	1.34	1.55	3.91	11.82	3.46
ESSH, T_8	1.291	1.33	1.59	3.93	11.64	3.63
ESSH, T_9	1.416	1.36	1.67	4.07	12.22	3.56
ESSH, T_{10}	1.531	**1.37**	**1.83**	**4.19**	**12.19**	**3.62**
Mamba-2 + FFN	1.526	2.93	3.66	7.17	18.96	15.18
Mamba-3 SISO + FFN	1.535	3.60	4.45	7.51	18.04	28.73
Mamba-3 MIMO + FFN	1.527	3.84	4.91	8.06	18.64	29.67
Transformer++	1.536	4.15	7.18	24.21	84.29	12.84

R.3 B300 PREFILL ACROSS THE TEN TIERS

The prefill comparison changes with context length. Transformer++ leads at 2K and remains faster than ESSH at 8K, but its throughput falls to 24.4 ktok/s at 128K while ESSH sustains 88.0, giving a $3.60\times$ advantage. Both models keep the same parameter count along their curves. The changed ordering therefore occurs as input length increases the full-attention workload, while the chunked parallel scan and fixed-window attention require work that grows linearly with context length.

This identifies the workload on which the hybrid's structure is useful: long inputs benefit from maintaining throughput as context length grows. The recurrent baselines also sustain their rates, with Mamba-2 reaching 222.7 ktok/s at 128K. Prefill computes many positions together, whereas decode repeatedly advances one state. ESSH's lowest decode latency and Mamba-2's highest prefill throughput describe different strengths under these two execution patterns.

Compact ESSH tiers remove channel and FFN work at every input position, but prefill peak memory decreases by a smaller proportion. The returned bf16 logits alone occupy 33.6 GB at 128K, independently of tier. Against that common output cost, the T_{10}–T_1 peak difference of 2.10 GB closely follows their 2.11 GB difference in weight storage. Keeping output allocation separate from state+KV explains why throughput, model size, and prefill peak memory have different relative changes.

Table 60: B300 batch-one full-logit prefill. Throughput: ktok/s; latency: ms; peak allocation: decimal GB. Full models use three sessions, with three warm-ups and five timed calls per session. Compact rows use single-session measurements. Full-model parameter counts differ by at most 0.34% from ESSH. Bold is ranked separately within each ruled group.

		Prefill ktok/s ↑				ms ↓	Peak GB ↓	
Model	Params (B)	2K	8K	32K	128K	2K	2K	128K
ESSH, T_1	0.474	75.0	**108.5**	**110.4**	**109.7**	27.3	**1.51**	**35.67**
ESSH, T_2	0.575	79.8	106.0	107.6	107.6	25.7	1.70	35.86
ESSH, T_3	0.716	77.9	101.9	103.2	103.4	26.3	1.99	36.15
ESSH, T_4	0.828	**81.4**	99.1	101.0	100.9	**25.2**	2.21	36.36
ESSH, T_5	0.931	74.6	96.8	98.4	98.5	27.5	2.42	36.57
ESSH, T_6	1.076	76.1	93.6	95.9	95.8	26.9	2.78	36.93
ESSH, T_7	1.168	74.0	92.2	94.2	93.9	27.7	2.98	37.13
ESSH, T_8	1.291	76.9	89.9	91.6	91.6	26.6	3.20	37.36
ESSH, T_9	1.416	76.1	88.1	90.0	89.8	26.9	3.41	37.56
ESSH, T_{10}	1.531	71.0	84.1	87.2	88.0	28.8	3.62	37.77
Mamba-2 + FFN	1.526	77.6	**216.9**	**223.6**	**222.7**	26.4	3.63	37.78
Mamba-3 SISO + FFN	1.535	76.6	98.9	102.2	102.2	26.7	3.63	37.78
Mamba-3 MIMO + FFN	1.527	50.4	51.7	52.4	52.3	40.6	**3.61**	**37.76**
Transformer++	1.536	**114.9**	118.3	67.9	24.4	**17.8**	3.62	37.78

R.4 370M ELASTIC FAMILIES ACROSS DEVICES

The 370M results test whether the deployment advantage persists at a smaller model scale. Full ESSH is faster than the near-size full MatFormer and MatMamba models on A100, H200, and B300. The H200 measurements give 0.910 ms for ESSH, versus 2.819 for MatFormer and 2.334 for MatMamba, corresponding to $3.10\times$ and $2.56\times$ faster decoding. ESSH also uses less state+KV, 23.11 versus 161.56 and 26.05 MB (Table 62). Each recorded shape uses three timing sessions at batch one after a 2K context. Each session uses 20 warm-ups and 100 timed steps per execution mode.

The models devote similar parameter budgets to different computations. MatFormer uses attention in every block, MatMamba has 48 recurrent blocks, and ESSH uses 22 blocks with three window layers and a fused recurrent path. Fewer recurrent invocations and the fused update are part of the complete ESSH deployment design. The observed advantage belongs to these executable model configurations at their stated sizes.

Within a family, truncation reduces parameters faster than latency because embeddings, residual dimensions, and the layer sequence remain. The tables therefore pair model size with measured response time at each tier. The A100/B300 tier measurements and H200 full-model comparison use the stated protocols.

Table 61: Batch-one decode at the 370M scale, with a 2K cache. (a) A100/B300 tier sweeps, with each family ranked internally before the full-model comparison. (b) H200 full-model repeat at the stated parameter counts. Values are three-session medians. ESSH state is fp32 and KV bf16; baselines retain native precision. Bold marks the best value within each comparison group and device/execution column. Equal tier labels across families need not have equal sizes.

(a) Tier sweeps on A100 and B300

Family	Tier	Params (M)	Graph ms ↓		Eager ms ↓	
			A100	B300	A100	B300
ESSH	T_1	92.1	**0.91**	**0.60**	**3.21**	**2.27**
ESSH	T_2	118.9	0.94	0.65	3.30	2.32
ESSH	T_3	154.7	1.01	0.68	3.40	2.43
ESSH	T_4	185.3	1.04	0.69	3.49	2.42
ESSH	T_5	214.5	1.10	0.68	3.49	2.46
ESSH	T_6	248.8	1.11	0.73	3.49	2.42
ESSH	T_7	274.3	1.13	0.74	3.51	2.47
ESSH	T_8	308.6	1.23	0.72	3.55	2.57
ESSH	T_9	337.9	1.24	0.73	3.59	2.56
MatFormer	T_1	130.9	**3.02**	2.83	13.71	9.34
MatFormer	T_2	146.0	3.14	**2.31**	**13.52**	8.92
MatFormer	T_3	168.7	3.21	2.90	**13.52**	9.26
MatFormer	T_4	191.5	3.29	2.38	13.60	**8.81**
MatFormer	T_5	214.2	3.27	2.93	13.68	9.28
MatFormer	T_6	248.2	3.34	3.00	13.70	9.30
MatFormer	T_7	270.9	3.37	2.42	13.61	8.88
MatFormer	T_8	301.2	3.39	2.43	13.63	8.99
MatFormer	T_9	331.5	3.44	2.43	13.67	8.95
MatMamba	T_1	83.3	**2.99**	**1.97**	**28.08**	18.58
MatMamba	T_2	102.4	3.21	2.27	28.45	19.43
MatMamba	T_3	131.1	3.03	2.07	28.09	**18.48**
MatMamba	T_4	159.7	3.19	2.38	28.61	19.59
MatMamba	T_5	188.4	3.18	2.19	28.19	18.83
MatMamba	T_6	226.6	3.24	2.20	28.24	18.63
MatMamba	T_7	255.3	3.36	2.49	28.56	19.67
MatMamba	T_8	293.5	3.44	2.56	28.40	19.54
MatMamba	T_9	331.7	3.44	2.58	28.37	19.55
ESSH	T_{10}	370.9	**1.31**	**0.74**	**3.59**	**2.52**
MatFormer	T_{10}	365.6	3.46	2.44	13.67	8.99
MatMamba	T_{10}	369.9	3.42	2.56	28.24	19.71

(b) H200 full models

Model	Params (M)	Graph ms ↓	Eager ms ↓	State+KV (MB) ↓
ESSH	370.9	**0.91**	**2.75**	**23.11**
MatFormer	365.6	2.82	10.31	161.56
MatMamba	369.9	2.33	19.14	26.05

R.5 CONSOLIDATED EFFICIENCY MEASUREMENTS

The consolidated table compares full models at the stated parameter budgets across three GPUs (Table 62). The 1.53B comparison fixes depth and residual width and matches parameter counts within 0.34%; the 370M comparison uses each trained architecture at a similar parameter count.

ESSH has the lowest recurrent-decoding latency in these full-model comparisons. Prefill has a different ordering: Mamba-2 is strongest among the recurrent models, while ESSH sustains higher long-context throughput than full attention. State+KV reflects another trade-off: ESSH's fixed window KV cache makes its total larger than Mamba-2's recurrent state, but substantially smaller than Transformer++'s full-context cache.

Table 62: Cross-device efficiency at the stated parameter counts. Graph decode uses a 2K cache; prefill uses batch one. Full-model rows use three-session medians at matched parameter counts. A100 MIMO uses a separate matched repeat; OOM is measured allocation failure. Bold ranks within each group and device. Memory is decimal; ESSH state is fp32 and KV bf16, with native baseline state precision.

Measurement / model	Params (M)	A100	H200	B300
Full models near 370M: graph decode, batch one (ms ↓)				
ESSH	370.9	**1.307**	**0.910**	**0.735**
MatFormer	365.6	3.464	2.819	2.443
MatMamba	369.9	3.416	2.334	2.562
Full models near 1.53B: graph decode, batch 1 (ms ↓)				
ESSH, T_{10}	1531.1	**3.06**	**1.67**	**1.37**
Mamba-2 + FFN	1526.0	4.44	2.91	2.93
Mamba-3 SISO + FFN	1534.9	5.09	3.52	3.60
Mamba-3 MIMO + FFN	1526.7	5.45	3.85	3.84
Transformer++	1536.3	5.87	4.15	4.15
Full models near 1.53B: graph decode, batch 128 (ms ↓)				
ESSH, T_{10}	1531.1	**9.44**	**5.35**	**4.19**
Mamba-2 + FFN	1526.0	12.49	7.81	7.17
Mamba-3 SISO + FFN	1534.9	16.70	8.85	7.51
Mamba-3 MIMO + FFN	1526.7	18.01	9.87	8.06
Transformer++	1536.3	46.83	30.03	24.21
Full models near 1.53B: graph decode, batch 512 (ms ↓)				
ESSH, T_{10}	1531.1	**31.93**	**17.63**	**12.19**
Mamba-2 + FFN	1526.0	38.63	22.05	18.96
Mamba-3 SISO + FFN	1534.9	53.26	24.12	18.04
Mamba-3 MIMO + FFN	1526.7	56.19	26.46	18.64
Transformer++	1536.3	OOM	OOM	84.29
Full-model prefill at 2K (ktok/s ↑)				
ESSH, T_{10}	1531.1	28.3	54.9	71.0
Mamba-2 + FFN	1526.0	**47.5**	73.8	77.6
Mamba-3 SISO + FFN	1534.9	32.5	71.8	76.6
Mamba-3 MIMO + FFN	1526.7	26.7	39.6	50.4
Transformer++	1536.3	39.0	**89.0**	**114.9**
Full-model prefill at 128K (ktok/s ↑)				
ESSH, T_{10}	1531.1	34.0	60.7	88.0
Mamba-2 + FFN	1526.0	**54.7**	**125.4**	**222.7**
Mamba-3 SISO + FFN	1534.9	37.7	83.9	102.2
Mamba-3 MIMO + FFN	1526.7	30.5	42.2	52.3
Transformer++	1536.3	11.0	18.4	24.4
Persistent state+KV at batch one, 2K (MB ↓)				
ESSH, T_{10}	1531.1	53.61	53.61	53.61
Mamba-2 + FFN	1526.0	**30.33**	**30.33**	**30.33**
Mamba-3 SISO + FFN	1534.9	59.64	59.64	59.64
Mamba-3 MIMO + FFN	1526.7	61.01	61.01	61.01
Transformer++	1536.3	469.99	469.99	469.99