INTERDOMAIN ATTENTION: BEYOND TOKEN-LEVEL KEY-VALUE MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Softmax attention-based transformers and recurrent memory-based models (e.g., state space models (SSMs)) form the foundation of modern sequence models. Hybrid architectures aim to combine the expressivity from attention and the computational efficiency from recurrent memory, yet so far common practices simply stack both attention and recurrent layers. We propose Interdomain Attention (IA) as a novel hybrid attention architecture that mixes attention and recurrent memory in the same layer. Motivated by a kernel regression view of softmax attention, IA compresses values and kernel features of keys into a recurrent memory parameterized by an SSM. Then IA approximates dot-product attention by implicitly reconstructing the attention matrix and values from memory states; importantly this construction maintains SSM's computational efficiency advantage. We demonstrate IA's efficacy by designing and pre-training IA network architectures, where ablation studies identify attention-like readout as the primary source of gain over SSMs. Across 125M-1.3B models pre-trained on text data, IA improves over an S4D baseline at matched memory size. It also outperforms softmax attention-based Transformer in perplexity and accuracy on common-sense reasoning tasks. Based on Caduceus backbone, IA matches or outperforms the original SSM-based architecture in both perplexity and downstream tasks on genomic data.

1 INTRODUCTION

The phenomenal success of GenAI and foundation models is mainly attributed to the invention of scalable sequence modeling architectures for long sequences, e.g., natural language and genomics. In this regard, softmax dot-product attention (Vaswani et al., 2017) and recurrent memory such as state space models (SSMs) (Gu et al., 2020; 2022a) are two representative paradigms for sequence model's building blocks. Though highly performant, softmax attention-based architectures are limited by their fundamental issue in computational and memory costs: given an input token sequence of length N, softmax attention incur $O(N^2)$ computational cost in FLOPs and $O(N)$ memory cost for the key-value (KV) cache required by low-latency inference (Shazeer, 2019; Dao et al., 2022; Dao, 2024). Methods such as hardware-aware computational kernels (Dao et al., 2022; Dao, 2024), speculative decoding (Leviathan et al., 2023) and distributed sharding (Shoeybi et al., 2020) have been developed to mitigate the issue, but they do not address the fundamental challenge of high computational and memory complexities.

On the other hand, SSMs and Mamba architectures (Gu & Dao, 2024; Dao & Gu, 2024) were proposed as a solution for the quadratic scaling issue. By maintaining a fixed-size recurrent state (which leads to $\mathcal{O}(1)$ memory cost), SSMs update their recurrent memory in $\mathcal{O}(1)$ time per token processing step, thereby reducing the computational cost to $\mathcal{O}(N)$. However, it pays the price of inferior performance to softmax attention architectures in tasks that require precise information recall from the input token sequences. This performance gap is largely addressed by hybrid architectures (Lenz et al., 2025; Glorioso et al., 2024; Ren et al., 2025; De et al., 2024; Fu et al., 2023) that combine both paradigms. In particular, recent frontier models, such as Evo2 (Brixi et al., 2026) for genomics or Qwen3.8-Next (Qiu et al., 2026) and Kimi-K3 (Team et al., 2026) for language, use building blocks that stack recurrent memory layers followed by softmax attention layers. Such practice makes it difficult to attribute the importance of each type of layer within the overall architecture, and it often requires extensive architecture search to find the best stacking strategy.

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.

In this paper, we propose Interdomain Attention (IA) as a novel hybrid attention architecture that mixes recurrent memory with dot-product attention in one layer. By interpreting softmax attention as computing predictions from a kernel regression model trained on key-value pairs (Tsai et al., 2019; Katharopoulos et al., 2020), IA represents this regression model's kernel with finite feature maps and then compresses key features and values in evolving recurrent memory states. Key features and values of the historical tokens are then reconstructed *implicitly* for dot-product attention; this implicit attention step maintains the computational efficiency of recurrent memory layers ($\mathcal{O}(1)$ computational cost per token) while leveraging the expressivity of dot-product attention operations.

Our methodological contributions include (1) the derivation of the IA framework (Section 3.1) and (2) its practical architecture design (Section 3.2). The headline results from the IA architecture include:

- On a 125M-1.3B language-modeling study on FineWeb-Edu at matched state size (Section 5.1), IA achieves better validation perplexity over softmax attention and an S4D baseline (Gu et al., 2022b) at every scale. At 1.3B IA surpasses softmax attention on seven out of eight commonsense reasoning tasks and on their average (Yang et al., 2025). At 1.3B, IA also achieves higher mean log-likelihood than S4D across 14 recall tasks.
- On a genomic modeling study (Section 5.2), by replacing the SSM component in the Caduceus backbone (Schiff et al., 2024) with IA, Caduceus-IA closely matches native Caduceus on the HG38 task, and its parameter-sharing (PS) variant Caduceus-IA-PS, which realizes reverse-complement (RC) equivariance, outperforms Caduceus-PS on 14 of 18 Nucleotide Transformer (Dalla-Torre et al., 2025) downstream tasks.

Ablation studies in both language and genomics tasks show that the attention-like, query-dependent readout of IA blocks is key to the improved performance when compared with SSM-based architectures. Complexity and inference run-time figures also show advantages of IA over softmax attention. All these evidences support IA as a promising hybrid architecture for future sequence models.

2 Background

Our architecture bridges kernel attention with fixed-size context compression in state space models. We review both frameworks below.

Attention as kernel regression Standard dot-product attention (Vaswani et al., 2017) projects input tokens $x_n \in \mathbb{R}^d$ into queries, keys, and values $q_n, k_n, v_n \in \mathbb{R}^d$, giving output for token i

$$o_i = \frac{\sum_{n=1}^{N} \mathcal{K}(q_i, k_n)\, v_n}{\sum_{n'=1}^{N} \mathcal{K}(q_i, k_{n'})}, \tag{1}$$

where $\mathcal{K}(q, k) = \exp(q^\top k/\sqrt{d})$ denotes the softmax kernel. As noted in prior work (Tsai et al., 2019; Katharopoulos et al., 2020; Choromanski et al., 2021; Wang et al., 2026a), this formulation corresponds to the Nadaraya–Watson kernel regression estimator (Nadaraya, 1964; Watson, 1964), where attention performs kernel regression over the key-value pairs $\{(k_n, v_n)\}_{n=1}^{N}$ with the kernel $\mathcal{K}$ and returns the kernel regression model's prediction given query input q_i for $i = 1, ..., N_q$.

State space models and HiPPO A linear continuous-time state space model (SSM) maps an input signal $z(t) \in \mathbb{R}$ to a latent state $u(t) \in \mathbb{R}^M$ via

$$\dot{u}(t) = A(t)\, u(t) + B(t)\, z(t). \tag{2}$$

The High-order Polynomial Projection Operators (HiPPO) framework (Gu et al., 2020) derives transition matrices $A(t) \in \mathbb{R}^{M\times M}$ and $B(t) \in \mathbb{R}^M$ that continuously maintain the projection coefficients of the input history onto M time-varying orthogonal polynomial bases $\{\phi_m^{(t)}\}_{m=1}^{M}$. This recurrence allows models to maintain a compressed representation of the context online, forming the foundation of modern deep SSMs (Gu et al., 2022a;b; Gu & Dao, 2024).

3 Interdomain Attention

We propose Interdomain Attention (IA) as a new design choice for attention mechanisms from kernel regression perspective. The IA architecture is inspired by HiPPO-SVGP (Chen et al., 2025) as an

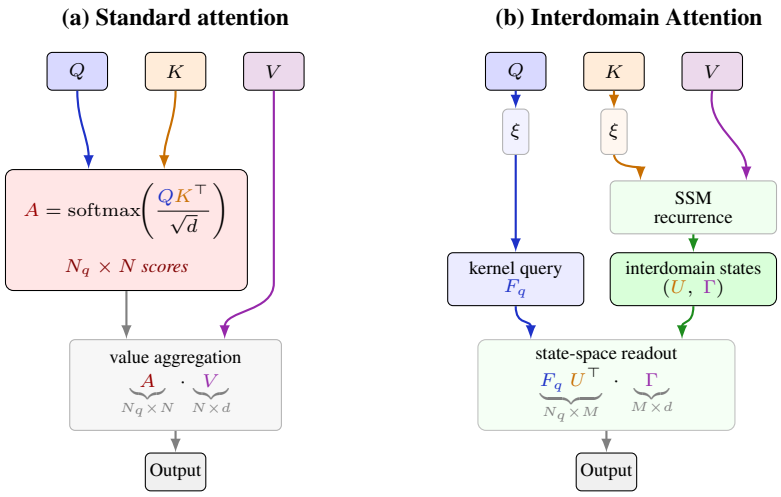


Figure 1: (a) Standard attention computes $N_q \times N$ scores from Q and K, then multiplies by V to produce the output. (b) Interdomain Attention maps queries and keys into a shared feature space to produce the kernel query matrix F_q ($N_q \times R$). Keys and values are compressed via SSM recurrence into M interdomain states: U ($M \times R$, key feature projections) and Γ ($M \times d$, value projections). State-space readout computes the output via the $N_q \times M$ interdomain attention matrix $F_q U^\top$.

instance of *interdomain* Gaussian process (Lázaro-Gredilla & Figueiras-Vidal, 2009), where memory of historical datapoints is constructed via functional projections of kernel features (which is referred to as "interdomain" kernel features). Therefore, instead of using an explicit KV cache, IA conducts attention with recurrent memory, by projecting the key features and values onto a function space formed by HiPPO/SSM basis functions (Gu et al., 2020) and computing attention therein. Figure 1 provides an overview of the IA framework, which is motivated by derivations in Section 3.1. The detailed design, including optional enhancements, are presented in Sections 3.2 and 3.3, respectively.

3.1 Implicit dot-product attention via interdomain kernel features

We first approximate the attention kernel with a finite feature representation $\xi(\cdot) \in \mathbb{R}^R$ to give $\mathcal{K}(q,k) \approx \xi(q)^\top \xi(k)$. Substituting this into Equation (1) gives

$$o_i \approx \hat{o}_i = \frac{\sum_{n=1}^{N} \xi(q_i)^\top \xi(k_n)\, v_n}{\sum_{n'=1}^{N} \xi(q_i)^\top \xi(k_{n'})}. \tag{3}$$

This formulation suggests the memory of keys should be constructed for their kernel features, motivating the following design of interdomain kernel features obtained by function projections. Specifically, we treat the keys and values as continuous functions of time, $k(t_n) := k_n$ and $v(t_n) := v_n$, and project the key features and the values onto the HiPPO basis $\{\phi_m^{(t_N)}\}_{m=1}^M$:

$$u_m^{(t_N)} = \int \xi\big(k(t)\big)\, \phi_m^{(t_N)}(t)\, dt, \quad \gamma_m^{(t_N)} = \int v(t)\, \phi_m^{(t_N)}(t)\, dt, \quad \eta_m^{(t_N)} = \int \phi_m^{(t_N)}(t)\, dt, \tag{4}$$

where $u_m^{(t_N)} \in \mathbb{R}^R$ and $\gamma_m^{(t_N)} \in \mathbb{R}^d$. Here $u_m^{(t_N)}$ is the interdomain kernel feature for the m-th temporal basis function after observing the N-th token (Chen et al., 2025), and we also refer to it as key coefficient. All three coefficients can be updated online with the HiPPO ODE Equation (2).

The key coefficients can then be used to reconstruct the key features as $\xi(k_n) \approx \sum_m u_m\, \phi_m(t_n)$. Substituting this reconstructed $\xi(k_n)$ into Equation (3) and swapping the sums over m and n leaves the token sums in the numerator as $\sum_n \sum_m u_m \phi_m(t_n)\, v_n = \sum_m u_m \sum_n \phi_m(t_n)\, v_n \approx \sum_m u_m \gamma_m$, where the third approximation comes from the HiPPO projection. Similarly for the denominator $\sum_n \sum_m u_m \phi_m(t_n) = \sum_m u_m \sum_n \phi_m(t_n) \approx \sum_m u_m \eta_m$. Dropping the superscript $^{(t_N)}$:

$$\hat{o}_i \approx \tilde{o}_i = \frac{\sum_{m=1}^{M} \big(\xi(q_i)^\top u_m\big)\, \gamma_m}{\sum_{m=1}^{M} \big(\xi(q_i)^\top u_m\big)\, \eta_m} \quad \Rightarrow \quad \tilde{O} = (F_q\, U^\top \Gamma) \oslash (F_q\, U^\top \eta), \tag{5}$$

with the query features $F_q = [\xi(q_1), ..., \xi(q_{N_q})]^\top \in \mathbb{R}^{N_q \times R}$, key coefficients $U = [u_1, ..., u_M]^\top \in \mathbb{R}^{M \times R}$, value coefficients $\Gamma = [\gamma_1, ..., \gamma_M]^\top \in \mathbb{R}^{M \times d}$ and $\eta = [\eta_1, ..., \eta_M]^\top \in \mathbb{R}^M$. Note that the sizes of the recurrent memory states U, Γ and η do not depend on N. In causal processing they become position-dependent: $U^{(i)}, \Gamma^{(i)}, \eta^{(i)}$ summarize tokens $1, \ldots, i$.

Compared with softmax attention, IA's key feature reconstruction is conducted *implicitly*, and IA's attention is conducted in a different domain. Here the query features F_q attend to the *interdomain* key kernel features U (instead of kernel features $[\xi(k_1), ..., \xi(k_N)]^\top$ as in softmax attention), which forms the "interdomain" attention matrix $F_q U^\top$. "Retrieval" is then performed on the function projections Γ of the original values. For practical design, we follow linear attention practices (Yang et al., 2024b) which drop the denominator in softmax attention and compute output as $F_q U^\top \Gamma$ only. A comparison between softmax attention and IA is visualized in Figure 1. IA is also different from SSM-based architecture due to the construction of recurrent memory that supports an attention-like readout.

3.2 A PRACTICAL IMPLEMENTATION OF INTERDOMAIN ATTENTION

We present a practical IA architecture design that instantiates the IA framework described above. The required design choices include kernel feature map, recurrent memory modules for function projections, and other considerations for numerical stability in training and inference.

Kernel feature map We use $\xi(\cdot) = \mathrm{L2Norm}\left(\mathrm{SiLU}\left(\mathrm{SC}(\cdot)\right)\right)$ proposed by Yang et al. (2024b), where $\mathrm{SC}(\cdot)$ is a causal depthwise 1D convolution applied to queries and keys before the head split, the short convolution used in recent subquadratic models (Fu et al., 2023; Poli et al., 2023; Gu & Dao, 2024), and $\mathrm{L2Norm}$ is the L2 normalization. $\xi(q)^\top \xi(k)$ is then a learned kernel, and we found this design numerically stable at large scale, corroborating reported results in Yang et al. (2024b).

Normalization To prevent state explosion and numerical instability in recurrent memory (Yang et al., 2024b), we apply $\mathrm{RMSNorm}$ with learnable biases to the key features and the values:

$$\tilde{k}_h^{(t)} = \mathrm{RMSNorm}\left(\xi(k_t^{(h)})\right) + b_h^k, \quad \tilde{v}_h^{(t)} = \mathrm{RMSNorm}(v_t^{(h)}) + b_h^v. \tag{6}$$

This forms the combined input $z_t^{(h)} = [\tilde{k}_h^{(t)}, \tilde{v}_h^{(t)}]$ to the recurrent memory discussed below.

Function projection using SSMs Per head h, we approximate the projections in Equation (4) with a complex diagonal S4D recurrence (Gu et al., 2022b) using $z_t^{(h)} \in \mathbb{R}^{R+d}$ after normalization:

$$\begin{aligned} s_t^{(h)} = \mathrm{diag}(\Lambda_h) s_{t-1}^{(h)} + B_h \, (z_t^{(h)})^\top, \quad \Lambda_h = \exp(\Delta_h A_h) \in \mathbb{C}^M, \; B_h = (\exp(\Delta_h A_h) - 1) \oslash A_h, \\ [U_h^{(t)}, \, \Gamma_h^{(t)}] = 2 \, \mathrm{Re}\Big(C_h s_t^{(h)}\Big) \in \mathbb{R}^{M \times (R+d)}, \quad C_h \in \mathbb{C}^{M \times M}. \end{aligned} \tag{7}$$

Here A_h adopts the diagonal S4D-Inv initialization (Gu et al., 2022b) approximating HiPPO-LegS dynamics, and A_h, Δ_h and C_h are learned end-to-end.

Implementation For efficient training, we adapt chunkwise execution from Flash Linear Attention (Yang et al., 2024a), combining parallel computation within each chunk with sequential state passing across chunks to keep total work linear in sequence length. We develop optimized chunkwise kernels in Triton for language-modeling scaling experiments and in TileLang (Wang et al., 2026b) for ablations (Appendix A.2, Appendix B). At inference time, the fixed-size recurrent state supports CUDA-graph capture and chunked prefill, enabling constant-memory token generation (Appendix F).

Complexity figures In training, IA achieves $\mathcal{O}(N)$ computational complexity in sequence length N, while standard softmax attention scales as $\mathcal{O}(N^2)$. Memory complexity for IA is $\mathcal{O}(N)$, as well as softmax attention with FlashAttention. During inference, softmax attention requires an expanding $\mathcal{O}(Nd)$ KV cache and $\mathcal{O}(Nd)$ computation per token generation. In contrast, IA compresses history tokens into recurrent states of size $\mathcal{O}(M(R+d))$ independent of N. Each decoding step takes $\mathcal{O}(M(R+d) + M^2)$ work, enabling constant-memory generation and CUDA-graph capture (decode latency and prefill memory benchmarks are reported in Appendix F). A detailed comparison of the complexity figures can be found in Table 1.

Table 1: Per-head computational and memory cost comparisons. FA denotes FlashAttention (Dao et al., 2022). For IA the figures are reported for full C_h, and K is the chunk length.

	Computational cost	Memory (inference)	Saved activations (training)
Softmax (FA)	$\mathcal{O}(N^2 d)$	$\mathcal{O}(Nd)$	$\mathcal{O}(Nd)$
IA (ours)	$\mathcal{O}(N[M(R+d)+M^2])$	$\mathcal{O}(M(R+d))$	$\mathcal{O}(\frac{N}{K}M(R+d)+N(R+d+M))$

3.3 Optional Architectural Enhancements

In addition to the base architecture, we explore optional refinements to enhance model expressivity without increasing the recurrent memory size. They are applied selectively to IA models trained on language and genomic sequence data.

Non-linear query projection While linear projection $q_t = W_q i_t$ given layer input $i_t \in \mathbb{R}^{d_{\text{model}}}$ is commonly used, improved query construction can also enhance the expressivity of the network layer. Hence we introduce an optional nonlinear query by adding a residual MLP to the projection:

$$q_t = W_q i_t + W_{q,2}\ \mathrm{SiLU}(W_{q,1} i_t). \tag{8}$$

We initialize $W_{q,2}$ to zero so that training begins from the standard linear projection.

Value paths and output gating We incorporate three refinements established in recent recurrent memory architectures (Gu et al., 2022b; Gu & Dao, 2024; Yang et al., 2024a). First, in trained IA models for genomics data we add a causal depthwise convolution followed by SiLU activation to construct the values $v_t^{(h)}$ before normalization. Second, we add a direct skip connection $D_h \odot v_t^{(h)}$ to the readout, initializing $D_h \in \mathbb{R}^d$ to ones. Finally, an output gate $g_t^{\text{out}} = \mathrm{SiLU}(W_g i_t)$ is applied per head before the final output merge across heads.

Selective writing Following practices in modern recurrent memory architectures (Gu & Dao, 2024), we introduce input-dependent gating to control the information for memory storage. We define a positive write gate $g_t^{\text{write}} = \exp\big[\alpha \odot \tanh\big(w \odot \mathrm{RMS}_0(\bar{k}_t) + b\big)\big]$, where $\bar{k}_t$ is kernel feature map for key just before L2Norm, RMS_0 is RMS normalization without learned parameters, and $w, b, \alpha \in \mathbb{R}^d$ are per-head learned parameters, with $R = d$. It scales both the key features and values before they enter the recurrence: $z_t^{\text{write}} = \big[\sqrt{g_t^{\text{write}}} \odot \tilde{k}_t,\ \sqrt{g_t^{\text{write}}} \odot \tilde{v}_t\big]$. We apply the same square-root gate for both key and value so that they are jointly modulated.

Selective memory decay Following Mamba (Gu & Dao, 2024), we dynamically adjust memory decay based on the input. For each mode with base transition $\Lambda_m = \rho_m \exp(\mathrm{i}\theta_m)$, an input-dependent $\delta_t > 0$ rescales the decay rate for memory modulation:

$$\Lambda_{t,m} = \rho_m^{\delta_t} \exp(\mathrm{i}\theta_m), \quad s_t = \Lambda_t \odot s_{t-1} + B\,(z_t^{\text{write}})^\top, \tag{9}$$

where, omitting the head index, $\delta_t = \exp\big[(\log 2)\tanh\big(w_\delta^\top z_t^{\text{write}} + b_\delta\big)\big]$, with learned per-head parameters w_δ and b_δ.

4 Related Work

Efficiency in attention mechanisms The quadratic cost of attention has motivated both faster implementations and alternatives to avoid explicit pairwise token interactions. At the implementation level, FlashAttention uses tiling to compute exact softmax attention without storing the full attention matrix, reducing GPU memory transfers while retaining quadratic arithmetic complexity (Dao et al., 2022). Hardware-efficient Linear Attention (see next paragraph for detailed discussion in linear attention), Flash Linear Attention (FLA), uses chunkwise parallel computation and kernel fusion to improve GPU utilization and reduce memory overhead (Yang et al., 2024a). Similarly, our IA also adopts hardware-efficient chunkwise execution. At the algorithmic level, Performer uses positive random features whose inner products approximate the softmax kernel with linear time and space complexity (Choromanski et al., 2021).

SSMs and Linear Attention SSMs and linear attention summarize history in a recurrent state whose size does not grow with sequence length, with state space duality formalizing connections between the two (Dao & Gu, 2024). SSMs describe the evolution of a latent state through structured dynamics: HiPPO compresses sequence history into a fixed-size memory through online basis projections (Gu et al., 2020), providing the foundation for S4, a structured state-space sequence model, and its diagonal variant S4D (Gu et al., 2022a;b), while Mamba introduces input-dependent state updates (Gu & Dao, 2024). Linear attention maintains key-value associations for query-dependent retrieval. Naive linear Transformers use finite feature maps to accumulate key–value statistics in a fixed-size state (Katharopoulos et al., 2020). More expressive linear attentions include Gated Linear Attention which controls memory retention (Yang et al., 2024a) and Gated DeltaNet which combines gating with delta-rule updates (Yang et al., 2025). Our IA builds on the projection view of memory to represent keys and values through projection coefficients corresponding to a shared temporal basis, and we combine these key and value representations at readout rather than accumulating their products.

Hybrid architectures Existing hybrid architectures combine recurrent or convolutional layers with attention mainly by stacking them: Jamba pairs Mamba layers with attention and uses mixture-of-experts modules in selected feedforward layers (Lenz et al., 2025), while Griffin combines gated linear recurrent units with sliding-window attention to compress history over a bounded cache of recent token representations for direct retrieval (De et al., 2024). Recent language models extend this approach: Qwen3.8-Next combines Gated DeltaNet layers with token-level attention, using Qwen Sparse Attention during continued pretraining (Qiu et al., 2026), while Kimi K3 combines Kimi Delta Attention with gated Multi-head Latent Attention (Team et al., 2026). In genomics, Evo 2 uses StripedHyena 2 to combine Hyena convolutional operators with attention layers (Brixi et al., 2026). H3 instead combines SSMs and multiplicative interactions within a layer, compressing key–value products for query-dependent retrieval (Fu et al., 2023). Our IA is a hybrid attention operator that integrates recurrent compression and content-based retrieval within a single layer, rather than stacking different layers.

5 Experiments

We evaluate the proposed IA architecture by pre-training and fine-tuning models on language and genomic data. The major baselines include SSM and softmax attention-based architectures, where our primary goal is to understand the benefits brought by the hybrid scheme of attention-like readout with recurrent memory. Architecture details can be found in Appendix A. In the main text we report summaries of the empirical results; detailed results can be found in Appendix.

5.1 Language modeling

Architecture and data We use Llama-3 backbone (Grattafiori et al., 2024) and replace the multi-head attention layer in pre-norm decoder with IA. We train four models with 125M, 350M, 760M, and 1.3B parameters on FineWeb-Edu dataset (Penedo et al., 2024), with Chinchilla-optimal token budgets of 2.5B, 7B, 15B, and 26B tokens respectively (Hoffmann et al., 2022). We use Llama-2 tokenizer (Touvron et al., 2023b), and context length 4096.

Scaling curve Figure 2(a) shows best-of-run validation perplexity for Softmax, S4D-only, and IA across four model scales on FineWeb-Edu. IA outperforms the matched-state S4D-only baseline at every scale. At the largest 1.3B scale, IA achieves a validation perplexity of 7.98, outperforming the Softmax baseline (8.62) and S4D-only (9.13). The precise numerical results and exact parameter counts are presented in Appendix Table 8 and Appendix A, respectively.

Downstream evaluation We evaluate the 1.3 B models via `lm-evaluation-harness` (Biderman et al., 2026) on the standard eight-task commonsense reasoning suite (Yang et al., 2025) , alongside language-modeling perplexities on LAMBADA and WikiText-2. As shown in Table 2, IA outperforms the Softmax baseline by $+3.03\%$ on the 8-task commonsense average and achieves significantly lower perplexity on both LAMBADA and WikiText-2. The S4D-only baseline underperforms both models. Details are provided in Appendix D.5. On associative recall, IA achieves higher mean answer log-likelihood than S4D-only for all 14 settings (Appendix D.8).

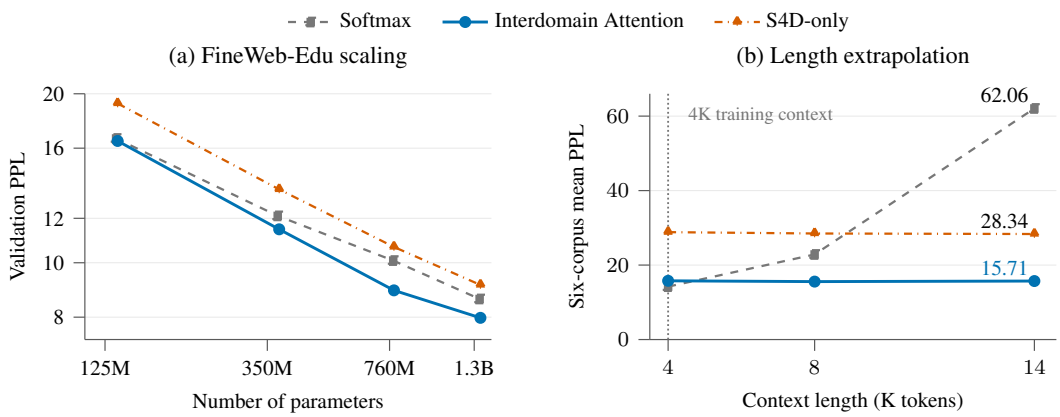


Figure 2: (a) Validation perplexity across model scales on FineWeb-Edu (exact values in Appendix Table 8). IA outperforms S4D-only at all scales and surpasses Softmax at 1.3 B. (b) Context length extrapolation up to 14K tokens at 1.3 B (per-corpus breakdowns in Appendix Table 14). IA remains flat with respect to length, whereas Softmax degrades sharply.

Table 2: LM downstream evaluation at 1.3 B. Commonsense is the eight-task average accuracy (percent); full per-task breakdowns are in Appendix Table 13.

Model	Commonsense ↑	LAMBADA PPL ↓	WikiText-2 PPL ↓
Softmax	51.51	21.03	20.55
S4D-only	50.23	41.02	25.27
Interdomain Attention	**54.54**	**14.76**	**19.77**

Length extrapolation IA avoids the positional-degradation issues inherent to attention with RoPE due to the use of recurrent memory. When evaluated at context lengths up to 14K ($3.5\times$ the 4K training context) across six long-context corpora (Figure 2(b)), IA at 1.3 B maintains nearly constant perplexity, whereas the Softmax baseline degrades sharply from 14.21 to 62.06 at 14K. Detailed result breakdown and associative-recall evaluations are in Appendices D.6 and D.7.

Inference speed We use CUDA-graph capture to measure IA's autoregressive decoding throughput. PyTorch SDPA with FlashAttention2 backend (Dao, 2024) is used for softmax attention. For short prefix length and small batch size, highly optimized SDPA keeps constant decode latency with respect to the prefix length. However, for long prefix length (e.g., 8k) and large batch size (e.g. 8), SDPA's latency start glowing while IA stays constant. Detailed speed results are reported in Appendix F.

5.2 Genomic Sequence Modeling

Architecture We use Caduceus backbone (Schiff et al., 2024) and replace its Mamba layers with IA, preserving its bidirectional processing and reverse-complement handling. We refer to *Caduceus-IA-PS* as the IA version of Caduceus that uses Parameter Sharing (PS) to preserve reverse-complement (RC) equivariance, which corresponds to Caduceus-PS, and *Caduceus-IA-Ph* as post-hoc (Ph) RC handling and data augmentation, which corresponds to Caduceus-Ph.

Pre-training We pre-train Caduceus-IA on the human reference genome (HG38) (Genome Reference Consortium, 2009) using bidirectional masked language modeling with the standard BERT masking recipe (Devlin et al., 2019). After 10,000 training steps at a context length of 1024, Caduceus-IA reaches a validation perplexity of 2.614, closely matching native Caduceus at 2.612 under the same context length and training budget (Appendix Table 16; see Appendix B.2 for baseline settings). We also pretrain a separate 7.7M parameters Caduceus-IA configuration with a context length of 131,072 tokens for 50,000 updates.

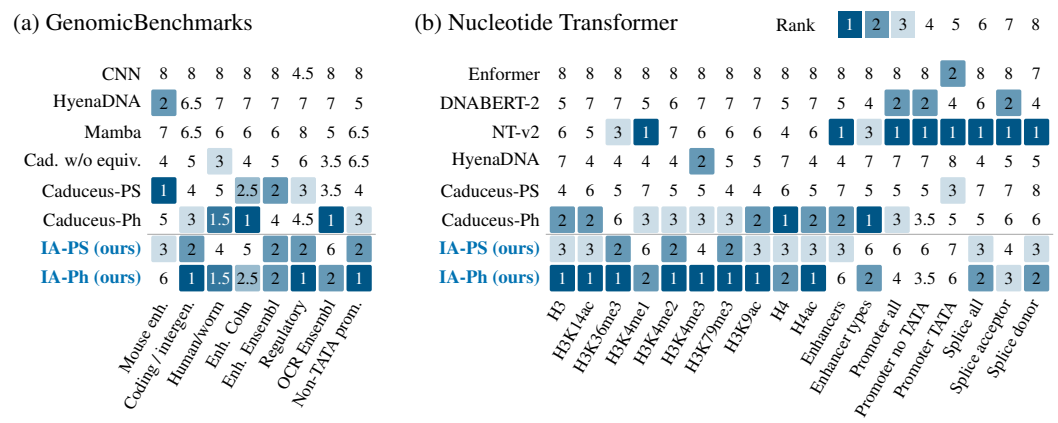


Figure 3: Task-wise model rankings on (a) GenomicBenchmarks and (b) Nucleotide Transformer (1 = best rank; top-3 shaded). Caduceus-IA variants rank among the top models across most tasks. Full scores and metrics are in Appendix Tables 17 and 18.

Downstream evaluation. We evaluate Caduceus-IA-PS and Caduceus-IA-Ph against published baselines (Schiff et al., 2024) on 8 GenomicBenchmarks regulatory element classification tasks (Grešová et al., 2023), 18 Nucleotide Transformer tasks such as histone marker prediction and splice site annotation (Dalla-Torre et al., 2025), and causal-eQTL variant-effect prediction (Avsec et al., 2021; Kao et al., 2024), which assesses whether SNPs affect gene expression. To match the Caduceus experiments, we use Caduceus-IA models pretrained at 1K context for GenomicBenchmarks (model dimension 118, 470K parameters) and Nucleotide Transformer (model dimension 256, 1.94M parameters), and at 131K context for variant-effect prediction (model dimension 256, 7.7M parameters). Here, model dimension refers to the dimension of each token's embedding.

Figure 3 summarizes the classification results through task-wise ranks, where detailed numerical results are provided in Appendix Tables 17 and 18. On **GenomicBenchmarks**, Caduceus-IA-Ph achieves the highest reported mean accuracy on three of eight tasks and shares the lead on two more, while published Caduceus baselines perform best on the remaining three. On **Nucleotide Transformer**, Caduceus-IA-PS and Caduceus-IA-Ph outperform their corresponding Caduceus baselines on 14 and 12 of 18 tasks, respectively. Caduceus-IA-Ph leads the full comparison on eight H3 and H4 tasks, while NT-v2 retains the lead in all promoter and splice-site tasks.

For **variant-effect prediction**, we group variants by distance to the nearest transcription start site. Caduceus-IA-PS outperforms HyenaDNA and NT-v2 in reported mean AUROC across all groups. Against their respective baselines, Caduceus-IA-PS matches performance at 0.679 within 30 kb, achieves a higher AUROC of 0.620 versus 0.612 at 30–100 kb, and a lower 0.574 versus 0.586 beyond 100 kb, while Caduceus-IA-Ph underperforms Caduceus-Ph across all groups (Appendix Table 19).

5.3 Analysis of the Interdomain Attention architecture

IA base architecture ablations To isolate the contributions of the base IA components, we separately ablate two core design choices. (1) For input structure, we compare how we make the feature z_t. In our base IA, before normalization of Equation (6), we use key feature $\xi(k_t)$ and value v_t. The baseline we use is linear projections $W_a k_t$ and $W_b v_t$ respectively. (2) For readout, we compare our attention-like, query-conditioned readout with linear readout (see Figure 6 in Appendix D.2). In ablations we use the base IA configuration (details in Appendix B.1). Table 3 shows that the attention-like, query-conditioned readout is the largest contributor to the improved performance. This justifies our core design of attention-like readout motivated by the IA framework which implicitly constructs approximations to dot-product attention.

Adding optional enhancements We next evaluate how each architectural enhancement methods introduced in Section 3.3 contributes to performance. Starting from an IA base model that omits short convolutions, we incorporate short convolutions to evaluate the effect of short convolution in the base IA, and added a residual query MLP, a direct skip connection for value, an output gate,

Table 3: Mechanism ablation with common initialization and final-checkpoint evaluation: validation NLL (↓). Non-selective/Selective denotes memory updates with selective writing and selective memory decay jointly disabled/enabled. Input preprocessing is described in Appendix B.1; additional studies with different evaluation settings are reported in Appendices B.3 and D.1.

		LM		Caduceus	
Input	Readout	Non-selective	Selective	Non-selective	Selective
Key feature/value	Query-conditioned	2.7949	2.7867	0.9893	0.9892
Key feature/value	Linear	3.0073	3.0002	1.0003	0.9939
Linear projections	Query-conditioned	2.8049	2.7964	0.9786	0.9783
Linear projections	Linear	2.9902	2.9820	0.9802	0.9788

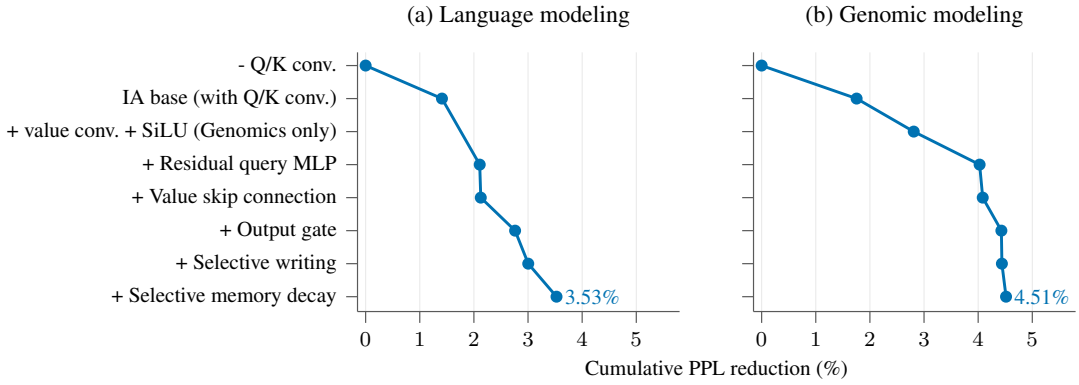


Figure 4: Cumulative perplexity reduction from sequentially adding architectural refinements in (a) language modeling and (b) genomic sequence modeling. (Details in Appendix Table 6)

selective writing, and selective memory decay in an incremental manner. In genomics experiments we additionally include depthwise convolution followed by SiLU, as mentioned in Section 3.3. Figure 4 shows that for language modelling most of the architectural enhancements, apart from the value skip connection and selective writing, provide decent improvements for the performance. For genomics, Q/K conv, value conv + SiLU activation and residual query MLP provide most of the gains. The final configurations reduce validation perplexity from 16.595 to 16.008 in language modeling and from 2.737 to 2.614 in genomic modeling.

6 Conclusion and Future Work

We propose Interdomain Attention (IA) that approximates dot-product attention by function projection of key features and values into a recurrent memory. At matched recurrent state size, base Interdomain Attention improves over an S4D baseline from 125M to 1.3B parameters; at 1.3B, it also outperforms softmax attention on validation perplexity and downstream common-sense reasoning tasks. Using the Caduceus backbone, IA-based architectures also show competitive or improved performance. Ablation studies across language and genomics identify the attention-like readout as the primary source of performance gains. For limitations, IA's training throughput is 2.8-times slower than that of PyTorch Flash SDPA, primarily due to less well-optimized GPU kernel when compared with FlashAttention2 (Appendix G). Future work includes building interdomain projections on more recent recurrent memories such as Mamba-3 and Gated DeltaNet instead of S4D, GPU kernel optimization, and formalizing a unified probabilistic framework to the connection between kernel attention and interdomain Gaussian processes (Chen & Li, 2023; Chen et al., 2025; Chen, 2026).

Reproducibility Statement

All details regarding algorithms and experiments are enumerated in this manuscript. In the event that the paper is accepted, we will open source model weights and code.

Ethics Statement

This work proposes a general-purpose sequence model architecture and evaluates it on text and genomic modeling tasks. It involves no human subjects research and no new data collection. All datasets are publicly available and used in accordance with their licenses. As with other foundation models for biological sequences, improved genomic sequence modeling is dual-use: it may advance the study of gene regulation and variant effects, but could in principle contribute to the design of harmful biological sequences.

AI Use Statement

The authors developed the core idea and mathematical derivations of Interdomain Attention without AI assistance. The authors used Codex, Claude Code, and Gemini to propose and test stabilization refinements and incremental ablations, suggest training settings, and assist with implementing Interdomain Attention, including GPU kernels and experimental pipelines. The tools also helped check the author-written main text and experiments for accuracy, polish the main-text wording, draft Appendix explanations from author outlines, create figures and tables, and identify potentially missing references. The authors checked GPU kernel outputs and gradients against a reference PyTorch implementation and verified AI-assisted text, figures, tables, and reported results against the implementation and experiment records. The authors take responsibility for the final content of this paper.

References

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 A. 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, 2024.

Žiga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R. Ledsam, Agnieszka Grabska-Barwinska, Kyle R. Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R. Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature Methods*, 18(10):1196–1203, October 2021. ISSN 1548-7105. doi: 10.1038/s41592-021-01252-x. URL https://doi.org/10.1038/s41592-021-01252-x.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, Charles Lovering, Niklas Muennighoff, Ellie Pavlick, Jason Phang, Aviya Skowron, Samson Tan, Xiangru Tang, Kevin A. Wang, Genta Indra Winata, François Yvon, and Andy Zou. Lessons from the Trenches on Reproducible Evaluation of Language Models, 2026. URL https://arxiv.org/abs/2405.14782.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about Physical Commonsense in Natural Language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, April 2020. doi: 10.1609/aaai.v34i05.6239.

Garyk Brixi, Matthew G. Durrant, Jerome Ku, Mohsen Naghipourfar, Michael Poli, Gwanggyu Sun, Greg Brockman, Daniel Chang, Alison Fanton, Gabriel A. Gonzalez, Samuel H. King, David B. Li,

Aditi T. Merchant, Eric Nguyen, Chiara Ricci-Tam, David W. Romero, Jonathan C. Schmok, Ali Taghibakhshi, Anton Vorontsov, Brandon Yang, Myra Deng, Liv Gorton, Nam Nguyen, Nicholas K. Wang, Michael T. Pearce, Elana Simon, Etowah Adams, Zachary J. Amador, Euan A. Ashley, Stephen A. Baccus, Haoyu Dai, Steven Dillmann, Stefano Ermon, Daniel Guo, Michael H. Herschl, Rajesh Ilango, Ken Janik, Amy X. Lu, Reshma Mehta, Mohammad R. K. Mofrad, Madelena Y. Ng, Jaspreet Pannu, Christopher Ré, John St. John, Jeremy Sullivan, Joseph Tey, Ben Viggiano, Kevin Zhu, Greg Zynda, Daniel Balsam, Patrick Collison, Anthony B. Costa, Tina Hernandez-Boussard, Eric Ho, Ming-Yu Liu, Thomas McGrath, Kimberly Powell, Sudarshan Pinglay, Dave P. Burke, Hani Goodarzi, Patrick D. Hsu, and Brian L. Hie. Genome modelling and design across all domains of life with Evo 2. *Nature*, 652(8112):1349–1361, April 2026. ISSN 1476-4687. doi: 10.1038/s41586-026-10176-5. URL https://doi.org/10.1038/s41586-026-10176-5.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Wenlong Chen. *Probabilistic Learning and Generation in Deep Sequence Models*. PhD thesis, Imperial College London, 2026.

Wenlong Chen and Yingzhen Li. Calibrating Transformers via Sparse Gaussian Processes. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=jPVAFXHlbL.

Wenlong Chen, Naoki Kiyohara, Harrison Zhu, Jacob Curran-Sebastian, Samir Bhatt, and Yingzhen Li. Recurrent Memory for Online Interdomain Gaussian Processes. 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. 134836–134873. Curran Associates, Inc., 2025. doi: 10.52202/085713-4499. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/c402501846f9fe03e2cac015b3f0e6b1-Paper-Conference.pdf.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking Attention with Performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

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 Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

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, 2018. URL https://arxiv.org/abs/1803.05457.

Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P. de Almeida, Hassan Sirelkhatim, Guillaume Richard, Marcin Skwark, Karim Beguir, Marie Lopez, and Thomas Pierrot. Nucleotide Transformer: building and evaluating robust foundation models for human genomics. *Nature Methods*, 22(2):287–297, February 2025. ISSN 1548-7105. doi: 10.1038/s41592-024-02523-z. URL https://doi.org/10.1038/s41592-024-02523-z.

Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/98ed250b203d1ac6b24bbcf263e3d4a7-Paper-Conference.pdf.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 2024.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359. Curran Associates, Inc., 2022. doi: 10.52202/068431-1189. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4599–4610, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.365. URL https://aclanthology.org/2021.naacl-main.365/.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Caglar Gulcehre. Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models, 2024. URL https://arxiv.org/abs/2402.19427.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Daniel Y. Fu, Tri Dao, Khaled Kamal Saab, Armin W. Thomas, Atri Rudra, and Christopher Re. Hungry Hungry Hippos: Towards Language Modeling with State Space Models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=COZDy0WYGg.

Genome Reference Consortium. Genome assembly GRCh37, 2009. URL https://www.ncbi.nlm.nih.gov/datasets/genome/GCF_000001405.13/.

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, 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, 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, May 2023. ISSN 2730-6844. doi: 10.1186/s12863-023-01123-8. URL https://doi.org/10.1186/s12863-023-01123-8.

Albert Gu and Tri Dao. Linear-Time Sequence Modeling with Selective State Spaces. In *Conference on Language Modeling*, 2024.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent Memory with Optimal Polynomial Projections. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1474–1487. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/102f0bb6efb3a6128a3c750dd16729be-Paper.pdf.

Albert Gu, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Albert Gu, Ankit Gupta, Karan Goel, 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., 2022b. doi: 10.52202/068431-2607. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/e9a32fade47b906de908431991440f7c-Paper-Conference.pdf.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030. Curran Associates, Inc., 2022. doi: 10.52202/068431-2176. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the Real Context Size of Your Long-Context Language Models? In *Conference on Language Modeling*, 2024.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient Attentions for Long Document Summarization. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1419–1436, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.112. URL https://aclanthology.org/2021.naacl-main.112/.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat After Me: Transformers are Better than State Space Models at Copying. 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. 21502–21521. PMLR, 2024.

Chia Hsiang Kao, Evan Trop, McKinley Polen, Yair Schiff, Bernardo P. de Almeida, Aaron Gokaslan, Thomas PIERROT, and Volodymyr Kuleshov. ADVANCING DNA LANGUAGE MODELS: THE GENOMICS LONG-RANGE BENCHMARK. In *ICLR 2024 Workshop on Machine Learning for Genomics Explorations*, 2024. URL https://openreview.net/forum?id=M3VlreGcC1.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, July 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA Reading Comprehension Challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023. URL https://aclanthology.org/Q18-1023/.

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.

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 Gerber, Elad Dolev, Eran Krakovsky, Erez Sa, 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 Meirom, Tal Delbari, Tal Ness, Tomer Asida, Tom Ben Gal, Tom Braude, Uriya Pumerantz, Joshua Cohen, Yonatan Belinkov, Yuval Globerson, Yuval Levy, and Yoav Shoham. Jamba: Hybrid Transformer-Mamba Language Models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 67959–67984, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a9ed43fa31dc8b4a7d7a673d713dcb5f-Paper-Conference.pdf.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast Inference from Transformers via Speculative Decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, July 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

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

Miguel Lázaro-Gredilla and Aníbal Figueiras-Vidal. Inter-domain Gaussian Processes for Sparse Inference using Inducing Features. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta (eds.), *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper_files/paper/2009/file/5ea1649a31336092c05438df996a3e59-Paper.pdf.

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.

E. A. Nadaraya. On Estimating Regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964. doi: 10.1137/1109020. URL https://doi.org/10.1137/1109020.

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 Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

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.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Re. Hyena Hierarchy: Towards Larger Convolutional Language Models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28043–28078. PMLR, July 2023. URL https://proceedings.mlr.press/v202/poli23a.html.

Zihan Qiu, Zekun Wang, Xiao Li, Yanpeng Li, Yang Xu, Yixuan Wang, Huaqing Zhang, Rui Men, Bochao Mao, Chengruidong Zhang, Fan Zhou, Hao Luo, Haofeng Huang, Haoran Lian, Haoyan Huang, Hongqing Chen, Jianwei Zhang, Jing Xu, Junjie Wang, Langshi Chen, Liangyu Wang, Linlang Jiang, Man Yuan, Minmin Sun, Peng Jin, Siqi Zhang, Siyu Wang, Xingzhang Ren, Yakai Wang, Yi Zhang, Yiming Dong, Yizhong Cao, Yubo Ma, Yunfei Mao, Bo Zheng, and Dayiheng Liu. On the Design of Qwen3.8-Next Architecture: Evaluation, Efficiency, and Training Stability, 2026. URL https://arxiv.org/abs/2608.30320.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

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.

Liliang Ren, Yang Liu, Yadong Lu, yelong shen, Chen Liang, and Weizhu Chen. Samba: Simple Hybrid State Space Models for Efficient Unlimited Context Language Modeling. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 53551–53575, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/84a7fc24ed52e8eff514c33e8ac76ea3-Paper-Conference.pdf.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8732–8740, April 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 Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/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.

Noam Shazeer. Fast Transformer Decoding: One Write-Head is All You Need, 2019. URL https://arxiv.org/abs/1911.02150.

Noam Shazeer. GLU Variants Improve Transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism, 2020. URL https://arxiv.org/abs/1909.08053.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified State Space Layers for Sequence Modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

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.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Guanduo Chen, Guangyu Chen, Guanzheng Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kexin Chen, Peng Chen, Ruijue Chen, Wentao Chen, Xin Chen, Yang Chen, Yanru Chen, Yifei Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Dazhi Cheng, Yean Cheng, Jialei Cui, Jingbing Cui, Anqi Dai, Jiaqi Deng, Hao Ding, Rui Ding, Shaofeng Ding, Mengfan Dong, Mengnan Dong, Yuhao Dong, Yuxin Dong, Angang Du, Chenzhuang Du, Dikang Du, Jusen Du, Yulun Du, Yu Fan, Jing Feng, Qiulin Feng, Yichen Feng, Kelin Fu, Qiang Fu, Fuxuan Gao, Hongcheng Gao, Jingyue Gao, Tong Gao, Weijia Gao, Shangyi Geng, Jie Gong, Linhu Gong, Shengao Gong, Xiaochen Gong, Qizheng Gu, Yicheng Gu, Shuhao Guan, Haiqing Guo, Shiqi Guo, Xiang Guo, Zhengyan Guo, Beixi Hao, Wenxin Hao, Xiaoru Hao, Dailan He, Haotian He, Lehan He, Qi He, Weiran He, Xinran He, Xinyi He, Yibo He, Yunjia He, Chao Hong, Tiange Hong, Hao Hu, Jiaxi Hu, Ruikun Hu, Weiming Hu, Yangyang Hu, Zhenxing Hu, Liang Hua, Jinbin Huang, Ke Huang, Ruiyuan Huang, Siying Huang, Weixiao Huang, Yan Huang, Zhengjie Huang, Zhiqi Huang, Yulong Hui, Chaobo Jia, Yutong Jiang, Zhejun Jiang, Zuoyou Jiang, Wenyi Jin, Xinyi Jin, Yu Jing, Huanjun Kong, Guokun Lai, Aidi Li, Cheng Li, Chengyuan Li, Cong Li, Fang Li, Guanyu Li, Haoyang Li, Jia Li, Junxiong Li, Lei Li, Letian Li, Lincan Li, Weihong Li, Wentao Li, Xintong Li, Yang Li, Yishen Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Zhengxiao Li, Zhiyuan Li, Jiawei Lin, Xiaohan Lin, Yibo Lin, Zichao Lin, Ziyan Lin, Bill Liu, Boxiao Liu, Chuan Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yipeng Liu, Zhengying Liu, Zhiheng Liu, Enzhe Lu, Haoyu Lu, Linqiang Lu, Tingzhan Lu, Zhiyuan Lu, Aotian Luo, G. Luo, Junyu Luo, Yifan Luo, B. Lyu, Wenzhou Lyu, Shaoguang Mao, Yuan Mei, Xin Men, Minqing Ni, Yixuan Niu, Siyuan Pan, Shujun Peng, Zhangyang Qi, Ruoyu Qin, ZeChao Qin, Zeyu Qin, Haiquan Qiu, Jianxin Qiu, Jiezhong Qiu, Bowen Qu, Yuhao Qu, Zeyu Shang, Youbo Shao, Han Shen, Jincheng Shi, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Wingchun Siu, Pengwei Song,

Xiaoxi Song, Jianlin Su, Yunfeng Su, Zhaochen Su, Lin Sui, Jingsong Sun, Junyao Sun, Shaoning Sun, Shuzhe Sun, Tongyu Sun, Yujun Sun, Yunpeng Tai, Chuning Tang, Heyi Tang, Sirui Tang, Zecheng Tang, Chaoran Tian, Rongpeng Tian, Yu Tian, Wei Tu, Chensi Wang, Chuang Wang, Chunjie Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hao Wang, Huaqing Wang, Hui Wang, Jiayi Wang, Jinglong Wang, Jinhong Wang, Jiuzheng Wang, Linian Wang, Shaobo Wang, Shenzhi Wang, Shuyi Wang, Si Wang, Siyuan Wang, Tianfu Wang, Wenjue Wang, Xingran Wang, Xinmei Wang, Xinyuan Wang, Xusheng Wang, Yalin Wang, Yangkun Wang, Yao Wang, Yaoyu Wang, Yejie Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhenhao Wang, Zhongsheng Wang, Zifan Wang, Chu Wei, Ming Wei, Shouxin Wei, Zichen Wen, Fan Wu, Haoning Wu, Rucong Wu, Wenhao Wu, Xiaoxue Wu, Yingcong Wu, Yongqi Wu, Yuxin Wu, Zijian Wu, Xinglang Xian, Chenxuan Xiang, Yuye Xiang, Bocheng Xiao, Chenjun Xiao, Xin Xiao, Jin Xie, Xiaotong Xie, Yifeng Xie, Zhe Xie, Bowei Xing, Yiming Xiong, Baosheng Xu, Boyu Xu, Jiale Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Qingtao Xu, Shuyao Xu, Suting Xu, Tiantian Xu, Tianxiang Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ye Xu, Yueni Xu, Ziyao Xu, Haonan Xue, Junjie Yan, Yaoyao Yan, Fan Yang, Guangyao Yang, Hao Yang, Junwei Yang, Ruoyu Yang, Wenjie Yang, Xiaofei Yang, Xinyu Yang, Yi Yang, Yiling Yang, Ying Yang, Yuchen Yang, Zhen Yang, Zhilin Yang, Zian Yang, Zuhao Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhanbo Ye, Bohong Yin, Haoxiang Yin, Xietong Yin, Chengzhen Yu, Haozhen Yu, Longhui Yu, Shengnan Yu, Shuying Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Tongtian Yue, Wei Yue, Yang Yue, Dunyuan Zha, Haobing Zhan, B. H. Zhang, Dehao Zhang, Fei Zhang, Hao Zhang, Haoyuan Zhang, Huanyu Zhang, Jiapei Zhang, Jiaxuan Zhang, Jin Zhang, Kaiyi Zhang, Miaozhen Zhang, Puqi Zhang, Qinglei Zhang, Rong Zhang, Rui Zhang, Shaoshuai Zhang, Shiyi Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yangkun Zhang, Ye Zhang, Yichi Zhang, Yikun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Zijing Zhang, Bin Zhao, Chenguang Zhao, Feifan Zhao, Jinglun Zhao, Jinxiang Zhao, Shuai Zhao, Wenshuo Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Haozhi Zheng, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Haofeng Zhong, Lei Zhong, Longguang Zhong, M. Zhou, Qiankang Zhou, Runjie Zhou, Ruozhang Zhou, Xinyu Zhou, Yiqiao Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yangjunfeng Zhu, Yuxuan Zhu, Zhen Zhu, Chen Zhuang, Weiyu Zhuang, and Xinxing Zu. Kimi K3: Open Frontier Intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models, 2023a. URL https://arxiv.org/abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023b. URL https://arxiv.org/abs/2307.09288.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer Dissection: An Unified Understanding for Transformer's Attention via the Lens of Kernel. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4344–4353, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1443. URL https://aclanthology.org/D19-1443/.

Lewis Tunstall, Leandro von Werra, and Thomas Wolf. *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. O'Reilly Media, Inc., 2022. ISBN 978-

1-0981-0324-8. URL https://huggingface.co/datasets/transformersbook/codeparrot.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *Journal of Machine Learning Research*, 27 (169):1–41, 2026a. URL http://jmlr.org/papers/v27/25-0903.html.

Lei Wang, Yu Cheng, Yining Shi, Zhiwen Mo, Zhengju Tang, Wenhao Xie, Tong Wu, Lingxiao Ma, Yuqing Xia, Jilong Xue, Fan Yang, and Zhi Yang. TileLang: Bridge Programmability and Performance in Modern Neural Kernels. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 73099–73134, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/76fb92288bf90360c527efb0d1c2aba6-Paper-Conference.pdf.

Geoffrey S. Watson. Smooth Regression Analysis. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, 26(4):359–372, 1964. ISSN 0581572X. URL http://www.jstor.org/stable/25049340.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated Linear Attention Transformers with Hardware-Efficient Training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 56501–56523. PMLR, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing Linear Transformers with the Delta Rule over Sequence Length. 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. 115491–115522. Curran Associates, Inc., 2024b. doi: 10.52202/079017-3668. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d13a3eae72366e61dfdc7eea82eeb685-Paper-Conference.pdf.

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.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/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.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. QMSum: A New Benchmark for Query-based Multi-domain Meeting Summarization. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5905–5921, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.472. URL https://aclanthology.org/2021.naacl-main.472/.

A Implementation and training details

A.1 Architecture details

For language modeling, we adopt a Llama-style pre-norm decoder backbone (Touvron et al., 2023a), implemented by adapting the Llama 3 reference code (Grattafiori et al., 2024), with RMSNorm (Zhang & Sennrich, 2019), SwiGLU feedforward layers (Shazeer, 2020), rotary position embeddings (RoPE) (Su et al., 2024) on the query/key inputs, untied embeddings, and no dropout. For genomic modeling, we build upon the Caduceus-PS backbone (Schiff et al., 2024), using bidirectional weight tying and reverse-complement parameter sharing (Appendix B.2). Scale-specific LM dimensions are given in Appendix A.3, and the ablation configurations in Appendix B.

The short convolution SC in section 3.2 has kernel size 4 in the language models and 8 in the genomic models; the genomic value convolution also has kernel size 8. A_h uses the S4D-Inv initialization (Gu et al., 2020; 2022b), $\mathrm{Re}(A_h) = -\frac{1}{2}$ and $\mathrm{Im}(A_h)(n) = \frac{M}{\pi}\big(\frac{M}{2n+1} - 1\big)$, and Δ_h is initialized log-uniformly in $[10^{-3}, 10^{-1}]$. The readout C_h is a full complex $M \times M$ matrix per head. The selective decay factor δ_t in Equation (9) lies in $[0.5, 2]$ and rescales only the modulus ρ_m of each mode; the phase θ_m is unchanged.

A.2 SSM kernel backends

The IA models in the language-modeling scaling study were trained using Triton chunkwise kernels with a chunk length of 64.

The sequential scan computes the S4D recurrence (Gu et al., 2022a;b) via a fused recurrence that is $\mathcal{O}(N)$ per input channel and state dimension, with parallelism over channels and state dimensions on GPU. The parallel scan uses the parallel prefix algorithm (Smith et al., 2023), achieving $\mathcal{O}(\log N)$ parallel depth. The chunkwise parallel scan, following the Flash Linear Attention scheme (Yang et al., 2024a), splits the sequence into chunks of size K. All chunks compute their terminal states in parallel, followed by a serial boundary propagation across N/K chunk boundaries, and a final parallel pass with corrected initial states. For the backward pass, both scan variants use segmented checkpointing with interval K: within each segment, previous states are recovered by inverting the diagonal SSM update (dividing by Λ_h). Loading the checkpoint at each segment boundary resets the numerical error that accumulates across this within-segment inverse recurrence.

A.3 Language modeling training

Backbone details. SwiGLU hidden dimensions are set to $\frac{2}{3} \cdot 4d$ rounded up to a multiple of 128. Residual-path output projections (w_o of attention and w_2 of SwiGLU) are re-initialized with standard deviation $0.02/\sqrt{2n_{\mathrm{layer}}}$ for n_{layer} layers, following the GPT-style residual scaling rule (Radford et al., 2019). Table 4 lists the per-scale dimensions. Softmax, S4D-only and IA share these dimensions at each scale; all use context length 4096, S4D state dimension $M{=}64$, and feature dimension $R{=}d_h$.

Table 4: Language-model dimensions per scale: model width d, number of layers n_{layer}, heads H, head dimension d_h ($= R$), and S4D state dimension M.

Scale	d	n_{layer}	H	d_h	M
125 M	768	12	12	64	64
350 M	1024	24	16	64	64
760 M	1536	24	16	96	64
1.3 B	2048	24	32	64	64

Parameter counts. We use the nominal model-size labels of GPT-3 (Brown et al., 2020). As detailed in Table 5, IA and S4D layers introduce less than 1% parameter overhead relative to the softmax baseline (ranging from 0.98% at 125 M to 0.52% at 1.3 B), primarily due to input RMSNorm parameters, per-head SSM dynamics, and local convolutions. Across all scales, IA and S4D parameter counts match to within 0.01%.

Table 5: Total trainable parameters for every cell in the language-modeling scaling sweep. Counts include token + output embeddings (untied, 32,000-vocab Llama 2 tokenizer (Touvron et al., 2023b)), all attention/SSM weights, the input-RMSNorm scales and biases of Equation (6), the layer-level RMSNorms, and the SwiGLU feedforward.

Model	125 M	350 M	760 M	1.3 B
Softmax	134,105,856	373,867,520	777,856,512	1,345,423,360
S4D-only	135,425,424	377,385,344	781,523,328	1,352,455,936
IA	135,416,208	377,360,768	781,498,752	1,352,406,784

Optimizer and schedule. Following Gu & Dao (2024): AdamW (Loshchilov & Hutter, 2019) with weight decay 0.1 and gradient clipping at 1.0; peak learning rate scaling with model size (3×10^{-3} at 125 M, 1.5×10^{-3} at 350 M, 1.25×10^{-3} at 760 M, and 1×10^{-3} at 1.3 B), with a 375 M-token linear warmup followed by cosine decay to 10^{-5} over the remainder of training; global token batch size $524{,}288$ per optimization step (sequence-packed); SSM input-normalization parameters (Equation (6)) and dynamics parameters (Δ_h, A_h) use a separate learning rate capped at 10^{-3} with weight decay disabled. The readout C_h follows the main learning-rate schedule with weight decay disabled. Training uses bfloat16 mixed precision (`torch.autocast`) with random seed 42, on Isambard-AI GH200 nodes with 4–32-way DDP. The final validation loss is within 0.01 nats of the best observed loss in 6 of the 12 scaling runs, with a maximum gap of 0.060 nats (IA, 760 M).

B ABLATION SETTINGS AND RESULTS

Execution backend. Across both language and genomic modeling, incremental ablations are run with custom TileLang (Wang et al., 2026b) forward and backward kernels (version 0.1.14) on NVIDIA GH200 GPUs. The kernels passed reference-equivalence tests for outputs and gradients. All reported incremental and mechanism ablations use a chunk length of 64 in both language and genomic modeling.

B.1 ABLATION SETTINGS AND PARAMETER COUNTS

The LM configurations use 12 layers of width 768, 4 heads, feature/value width 192, 64 complex state modes and query-MLP hidden width 317. The genomic configurations use 4 layers of width 256, 16 heads, feature/value width 16, 8 complex modes and query hidden width 266. Q/K short convolutions have width 4 in LM and 8 in DNA; the DNA V convolution also has width 8. The query MLP output projection is initialized to zero. All configurations use the same initial values for their shared parameters. In configurations where Q/K convolution is absent (such as the ablation start), Q/K SiLU activations remain enabled.

LM final validation uses 80 fixed batches of 8 sequences and evaluation seed 20260911. HG38 masked LM uses 15% masking and fixed validation mask seed 0. All ablations use training seed 42 and the full update budgets stated in the main text.

Table 6: Incremental ablations: final fixed-validation perplexity and parameter counts, seed 42. Additions accumulate within each panel; ΔNLL is relative to the preceding applicable configuration. The value convolution is used only for HG38 (dashes denote not applicable). Shared initial tensors and recurrent-state dimensions are fixed within each domain.

Language modeling			
Cumulative addition	Parameters	PPL ↓	ΔNLL ↓
Base IA without Q/K short convolution	134,528,304	16.595	—
+ Q/K short convolution (Base IA)	134,620,464	16.361	-0.0142
+ Value convolution + SiLU	—	—	—
+ Residual query MLP	140,463,408	16.244	-0.0071
+ Value skip connection	140,472,624	16.240	-0.0002
+ Output gate	147,550,512	16.135	-0.0065
+ Selective writing	147,578,160	16.095	-0.0025
+ Selective memory decay	147,596,640	16.008	-0.0054
HG38 masked language modeling			
Cumulative addition	Parameters	PPL ↓	ΔNLL ↓
Base IA without Q/K short convolution	1,076,864	2.737	—
+ Q/K short convolution (Base IA)	1,113,728	2.689	-0.0177
+ Value convolution + SiLU	1,122,944	2.661	-0.0108
+ Residual query MLP	1,667,712	2.627	-0.0126
+ Value skip connection	1,668,736	2.626	-0.0006
+ Output gate	1,930,880	2.616	-0.0036
+ Selective writing	1,933,952	2.616	-0.0001
+ Selective memory decay	1,936,064	2.614	-0.0008

Mechanism-ablation input processing. In the key feature/value configurations, the key projection passes through a short convolution and SiLU followed by L2 normalization, and the value projection is not convolved. In the linear-projection configurations, both input halves pass through a short convolution and SiLU without L2 normalization. Both retain the per-half RMSNorm and bias of Equation (6). Query-conditioned configurations apply short convolution, SiLU and L2 normalization to the query; linear-readout configurations have no query path. The comparison between input types therefore includes these preprocessing differences, whereas the two readouts are compared under identical input processing. The value convolution and SiLU of Section 3.3 are not applied.

B.2 NATIVE CADUCEUS BASELINE SETTINGS

The native baseline has 4 layers of width 256, expansion factor 2, state dimension 16 and 1,934,592 parameters, with bidirectional weight tying and reverse-complement parameter sharing. Its final checkpoint is taken at update 10,000. Like the Caduceus-IA incremental ablation, it uses HG38, context length 1024, 15% masking, training seed 42, 1024 sequences per update, and fixed-mask validation with mask seed 0 over 100 batches of 128 sequences. Both use BF16 with TF32 disabled, peak learning rate 0.008, 1,000 warmup updates, cosine decay to 0.0001 and weight decay 0.1. The native model uses its original initialization and standard BF16 residual accumulation, whereas Caduceus-IA enables FP32 residual accumulation for numerical stability and adopts shared initialization across its ablation configurations.

B.3 CADUCEUS-IA MECHANISM ABLATION WITH A MINIMAL INTERFACE

Table 7 repeats the Caduceus-IA mechanism ablation without the residual query MLP, value convolution, direct value skip connection and output gate of Section 3.3.

Table 7: Caduceus-IA mechanism ablation with a minimal interface: final validation NLL and parameter counts at 1K context after 10,000 updates, seed 42. Non-selective and Selective denote memory updates with selective writing and selective memory decay both disabled and both enabled, respectively.

		Non-selective		Selective	
Input	Readout	NLL ↓	Parameters	NLL ↓	Parameters
Key feature/value	Query-conditioned	0.9903	1,113,728	0.9896	1,118,912
Key feature/value	Linear	1.0099	1,096,320	0.9991	1,101,504
Linear projections	Query-conditioned	0.9792	1,132,160	0.9796	1,137,344
Linear projections	Linear	0.9847	1,114,752	0.9806	1,119,936

C LANGUAGE-MODELING SCALING AND DOWNSTREAM DETAILS

Setup. We pretrain Llama-style decoder-only models (Touvron et al., 2023a) at four scales (125 M, 350 M, 760 M, 1.3 B parameters) on the FineWeb-Edu corpus (Penedo et al., 2024) with the Llama 2 tokenizer (Touvron et al., 2023b) (32K vocabulary) and a training context length of 4096. Each scale is trained at its Chinchilla-optimal token budget (Hoffmann et al., 2022) (approximately $20\times$ tokens per parameter): 2.5, 7, 15, and 26 billion tokens, respectively. We follow the training recipe of Gu & Dao (2024); full optimizer, schedule, and hardware details are in Appendix A.3. We report best-of-run validation perplexity from the cosine-decay schedule.

We compare three main conditions and an additional 125 M RoPE control, all sharing the same Llama-style backbone, dataset, tokenizer, and training recipe; only the attention or recurrent layer in each block differs:

- *Softmax*: canonical multi-head softmax attention with rotary position embeddings;
- *IA*: the mechanism of Section 3.2;
- *S4D-only*: an S4D control sharing the complex S4D core of IA (full per-head $C_h \in \mathbb{C}^{M\times M}$) but with both primary components removed: the key feature/value input $[\xi(k_t), v_t]$ is replaced by a linear-projection input $[a_t, b_t]$, and the query-conditioned readout is replaced by a learned linear contraction of the SSM coefficients;
- *S4D-only + RoPE*: the S4D-only control with rotary position embeddings applied to the first input half (a_t), tested at 125 M only as a RoPE control.

Additional mechanism and RoPE controls at 125 M. Additional 125 M ablations examine key feature/value input, query-conditioned readout, and RoPE at matched recurrent-state size, using best-of-run validation loss (Appendix D.1); Appendix D.2 shows the per-condition data flow.

State budget. IA and the S4D-only variants have the same recurrent-state size at every scale by construction (Appendix D.3).

Table 8: Language-modeling scaling: best-of-run validation PPL (↓) on FineWeb-Edu. Parameter counts closely match across models at each nominal scale (Table 5).

Nominal size	125M	350M	760M	1.3B
Training tokens	2.5B	7B	15B	26B
Softmax	16.65	12.12	10.09	8.62
S4D-only	19.22	13.51	10.67	9.13
IA	**16.48**	**11.47**	**8.93**	**7.98**

Scaling. Figure 5 plots best-of-run FineWeb-Edu validation perplexity for the three main conditions across the four scales. In the controlled comparison with S4D-only at matched recurrent-state size, IA reaches 13–16% lower validation perplexity at every scale. Against the same-recipe softmax reference, IA achieves about 1% lower perplexity at 125 M and 7.4% lower perplexity at 1.3 B (7.98 vs. 8.62).

Downstream evaluation at 1.3 B. We evaluate the 1.3 B Softmax, S4D-only, and IA models via `lm-evaluation-harness` (Biderman et al., 2026) on the 8-task commonsense protocol of Yang et al. (2025) (LAMBADA (Paperno et al., 2016), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), ARC-e/ARC-c (Clark et al., 2018), SIQA (Sap et al., 2019), BoolQ (Clark et al., 2019)), together with LAMBADA and WikiText-2 (Merity et al., 2017) language-modeling perplexities; aggregate and per-task results are in Appendices D.4 and D.5. Relative to the same-recipe softmax baseline, IA improves the commonsense 8-task average by 3.03 percentage points and reduces bits per byte (BPB) by 0.010 on WikiText-2 and by 0.0764 on LAMBADA. The S4D control is 1.27 percentage points lower on commonsense, has approximately twice the LAMBADA perplexity (41.02 vs. 21.03), and has 6% higher validation perplexity than Softmax.

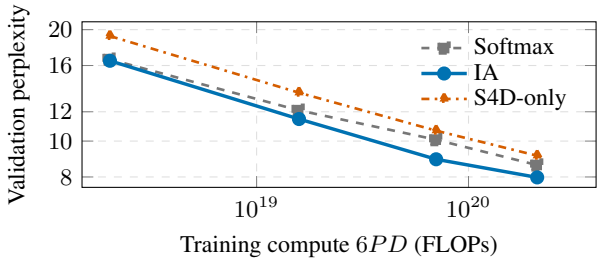


Figure 5: The same scaling results as Figure 2(a), plotted against estimated training compute. FineWeb-Edu validation perplexity vs. training compute (log–log scale), $6PD$ FLOPs for total parameters P and total training tokens D (Chinchilla convention (Hoffmann et al., 2022)). Each point is best-of-run perplexity at the Chinchilla-optimal token budget for that scale (2.5, 7, 15 and 26 B tokens for 125 M, 350 M, 760 M, 1.3 B parameters).

Length extrapolation. The fixed-state structure that IA inherits from S4D summarizes the entire prefix in a state of size independent of context length, so the recurrence has no built-in dependence on training length. Our RoPE-based softmax baseline, by contrast, degrades rapidly outside its training context. We evaluate the 1.3 B Softmax and IA models at $N \in \{4\text{K}, 8\text{K}, 14\text{K}\}$ on six long-context corpora: PG19 (Rae et al., 2020), CodeParrot (Tunstall et al., 2022), GovReport (Huang et al., 2021), NarrativeQA (Kočiský et al., 2018), Qasper (Dasigi et al., 2021), and QMSum (Zhong et al., 2021). Within the 4 K training context, Softmax has lower average perplexity than IA (14.21 vs. 15.77). At 8 K and 14 K, Softmax perplexity increases by factors of 1.6 and 4.4, respectively, whereas IA remains within 0.25 of its 4 K value. The S4D-only control is also length-flat (Table 9).

Table 9: Length-extrapolation perplexity at context length N, averaged over six long-context corpora (PG19, CodeParrot, GovReport, NarrativeQA, Qasper, QMSum). Training context is 4 K; 8 and 14 K are out-of-distribution. The per-corpus matrix is in Appendix D.6.

Context length	4 K (train)	8 K	14 K
Softmax	**14.21**	22.85	62.06
S4D-only	28.84	28.48	28.34
IA	15.77	**15.56**	**15.71**

Associative recall evaluations. On RULER single-needle-in-a-haystack (Hsieh et al., 2024), Phonebook retrieval (Jelassi et al., 2024), and the Based zero-shot recall suite (Arora et al., 2024) at 1.3 B (Appendix D.7), Softmax achieves higher accuracy within the 4 K training context in every setting except Based SQuAD. IA exceeds S4D-only on RULER and all six Based tasks, while both models have zero exact-match accuracy on Phonebook. On RULER at 8 K, IA achieves 3.5% accuracy versus 0% for Softmax and S4D-only. In the complementary teacher-forced evaluation, IA achieves higher mean reference-answer log-likelihood than S4D-only in all 14 settings (Appendix D.8). On LongBench-14 (Bai et al., 2024), Softmax and IA achieve comparable performance (0.1240 vs. 0.1229), with both outperforming S4D-only (0.0857; Appendix D.4).

D SUPPLEMENTARY LANGUAGE-MODELING RESULTS

D.1 ADDITIONAL MECHANISM AND ROPE CONTROLS AT 125 M

The two core architectural components of IA define two binary axes: (i) the *key feature/value input* $z_t = [\xi(k_t), v_t]$, which partitions the SSM input into key-feature and value halves, and (ii) the *query-conditioned readout* $\xi(q_t)\, U_t^\top\, \Gamma_t$, which enables per-token query interaction with the compressed recurrent coefficients. Rotary position embedding (RoPE) provides an additional control. Table 10 reports six configurations alongside the softmax baseline; Figure 6 illustrates the corresponding computational graphs.

Table 10: Mechanism decomposition at 125 M / 2.5 B tokens. "vs Softmax" is the relative change in validation perplexity (positive = worse than softmax). Val. loss is best-of-run; Val. perplexity is $\exp(\text{Val. loss})$.

Model	Val. loss	Val. perplexity	vs Softmax
Base IA	**2.8020**	**16.48**	**−1.02%**
Base IA, no RoPE	2.8038	16.51	−0.84%
Softmax	2.8122	16.65	N/A
Linear-projection input + query-conditioned readout	2.8298	16.94	+1.78%
S4D-only	2.9561	19.22	+15.48%
S4D-only + RoPE	2.9844	19.77	+18.79%
Key feature/value input + linear readout	3.0048	20.18	+21.23%

Ablation analysis. At matched recurrent-state size, IA reduces validation perplexity from 19.22 for S4D-only to 16.48, a 14.3% relative reduction. Replacing the key feature/value input with a linear-projection input $[a, b]$ raises perplexity to 16.94, within 0.3 perplexity of softmax. By contrast, retaining the key feature/value input but removing the query-conditioned readout raises perplexity to 20.18. Removing Q/K RoPE changes perplexity from 16.48 to 16.51. These results suggest that the query-conditioned readout is the primary driver of performance at this scale and that RoPE has a negligible effect.

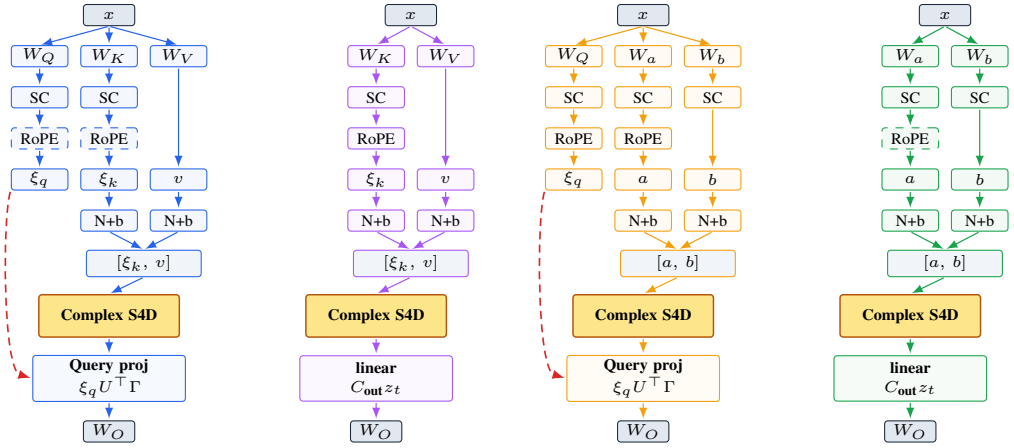
D.2 MECHANISM-CELL ARCHITECTURE DIAGRAM

D.3 STATE BUDGET AT 1.3 B

Table 11 summarizes the recurrent-state size of the evaluated layers at the 1.3 B scale, following the state-DoF accounting convention of Lahoti et al. (2026, Prop. 2): a complex SSM state of dimension n counts as $2n$ real DoF. The SSM in IA ingests a two-half input $z_t = [\xi(k_t), v_t]$ of width $d_{\text{feat}} + d_h$ ($= 2d_h$ in our configuration, where $d_{\text{feat}} = d_h$), so the per-cell state is $2(d_{\text{feat}} + d_h)M$ real DoF; the S4D-only variants likewise ingest a two-half $[a, b]$ input of the same width and inherit the same per-cell state. With matched head counts H and matched M, IA and the S4D-only variants have identical state size by construction. The Softmax KV cache grows linearly with sequence length N.

Table 11: Recurrent-state size at 1.3 B (d=2048, 24 layers, H=32, d_h=64, M=64). #cells is the number of independent state cells per layer; per-cell DoF is the size of each cell's hidden state in real-valued degrees of freedom; total DoF is their product, reported per layer. S4D-only + RoPE has the same state size as S4D-only.

Model	#cells	per-cell DoF	total DoF
Softmax	N/A	KV cache (grows with N)	$4{,}096\,N$
S4D-only	$H = 32$	$2(d_h{+}d_h)\,M = 16{,}384$	524,288
IA	$H = 32$	$2(d_{\text{feat}}{+}d_h)\,M = 16{,}384$	524,288



(a) Interdomain Attention

(b) Key feature/value input, linear readout

(c) Linear projections input, query-conditioned readout

(d) S4D-only (dashed: optional RoPE)

Figure 6: Architecture across the four mechanism cells. **(a)** *IA* uses both the key feature/value input $z_t = [\xi(k_t), v_t]$ and the query-conditioned readout $\xi(q_t)U_t^\top \Gamma_t$. **(b)** *Key feature/value input, linear readout* retains the key feature/value input but replaces the query-conditioned readout with a learned linear contraction. **(c)** *Linear-projection input, query-conditioned readout* feeds the SSM a linear-projection input $[a_t, b_t]$. **(d)** *S4D-only* ablates both components (dashed lines indicate the optional RoPE branch evaluated in Table 10). Branches drawn in the subfigure color are active; the dashed red curve marks the query bypass; the amber complex-valued S4D core is identical across all four variants. The two "N+b" boxes denote the per-head input RMSNorm and learnable bias of Equation (6), applied independently to each SSM input half.

D.4 Downstream evaluation at 1.3 B

Table 12 reports validation loss and downstream evaluation metrics for the three 1.3 B runs. LongBench (Bai et al., 2024) uses the 14-subtask configuration of Yang et al. (2025), with scores normalized to $[0, 1]$.

Table 12: Validation cross-entropy and downstream metrics for the 1.3 B / 26 B-token runs. LAMBADA (Paperno et al., 2016) and WikiText-2 (Merity et al., 2017) are evaluated zero-shot via `lm-evaluation-harness` (Biderman et al., 2026). Accuracy is reported in percent; LongBench scores are on $[0, 1]$. BPB (bits per byte) is the total negative log-likelihood in bits divided by the corresponding UTF-8 byte count.

Metric	Softmax	S4D-only	IA
Validation CE ↓	2.155	2.212	**2.077**
Commonsense 8-task avg (%) ↑	51.51	50.23	**54.54**
LAMBADA accuracy (%) ↑	39.43	25.64	**44.36**
LAMBADA perplexity ↓	21.03	41.02	**14.76**
LAMBADA BPB ↓	0.6578	0.8021	**0.5814**
WikiText-2 perplexity ↓	20.55	25.27	**19.77**
WikiText-2 BPB ↓	0.8156	0.8713	**0.8051**
LongBench 14-task avg ↑	**0.1240**	0.0857	0.1229

Table 13: Per-task accuracy (%, higher is better) of the eight-task commonsense suite of Yang et al. (2025), evaluated on the 1.3 B / 26 B-token runs via `lm-evaluation-harness` (Biderman et al., 2026). HellaSwag and ARC-c use length-normalized accuracy; all others use plain accuracy. The 8-task average corresponds to the aggregate reported in Table 12.

Task	Softmax	S4D-only	IA
LAMBADA (acc)	39.43	25.64	**44.36**
PIQA	69.42	70.67	**71.38**
HellaSwag (acc_norm)	48.53	49.34	**54.27**
WinoGrande	54.22	53.83	**57.85**
ARC-easy	65.87	67.55	**69.91**
ARC-challenge (acc_norm)	33.02	33.36	**37.46**
SIQA	40.28	41.10	**41.45**
BoolQ	**61.28**	60.37	59.63
8-task avg	51.51	50.23	**54.54**

Table 14: Per-corpus validation perplexity at context lengths $N \in \{4, 8, 14\}$ K for the 1.3 B Softmax, IA, and S4D-only models on PG19 (Rae et al., 2020), CodeParrot (Tunstall et al., 2022), GovReport (Huang et al., 2021), NarrativeQA (Kočiský et al., 2018), Qasper (Dasigi et al., 2021), and QMSum (Zhong et al., 2021). Training context is 4 K; 8 K and 14 K are out-of-distribution. Table 9 reports the six-corpus average.

Corpus	Model	4 K	8 K	14 K
PG19	Softmax	16.34	26.78	70.95
	S4D-only	19.96	19.75	19.66
	IA	**15.86**	**15.73**	**15.94**
CodeParrot	Softmax	**12.13**	22.23	74.85
	S4D-only	53.10	52.44	52.16
	IA	17.62	**17.31**	**17.48**
GovReport	Softmax	**7.33**	12.20	31.76
	S4D-only	9.19	9.07	9.01
	IA	7.36	**7.27**	**7.29**
NarrativeQA	Softmax	**21.28**	34.36	92.75
	S4D-only	36.83	36.42	36.26
	IA	22.85	**22.67**	**22.99**
Qasper	Softmax	**14.43**	22.94	60.08
	S4D-only	23.53	23.22	23.13
	IA	15.75	**15.52**	**15.63**
QMSum	Softmax	**13.74**	18.57	41.96
	S4D-only	30.44	29.97	29.80
	IA	15.20	**14.88**	**14.94**

D.5 COMMONSENSE 8-TASK BREAKDOWN AT 1.3 B

D.6 PER-CORPUS LENGTH-EXTRAPOLATION PERPLEXITY

D.7 RECALL-HEAVY TASKS AT 1.3 B

Table 15 reports both generation accuracy and reference-answer log-likelihood on the same recall settings. Likelihood scoring is described in Appendix D.8.

Table 15: Recall evaluation of the same three 1.3 B checkpoints: generation accuracy and teacher-forced reference-answer log-likelihood (nats per answer). RULER (Hsieh et al., 2024) uses S-NIAH-1; Phonebook (Jelassi et al., 2024) uses exact match; Based (Arora et al., 2024) uses case-insensitive contains accuracy. Each synthetic setting has 200 examples; all examples in each Based validation split are scored for likelihood. $\Delta_{\mathrm{IA-S4D}}$ is the mean paired log-likelihood difference. Bold marks the highest value for each metric within each setting.

	Accuracy (%) ↑			Answer log-likelihood ↑			
Setting	Softmax	S4D-only	IA	Softmax	S4D-only	IA	$\Delta_{\mathrm{IA-S4D}}$
RULER S-NIAH-1 (passkey retrieval)							
1K context	**99.50**	0.00	65.50	−2.14	−18.16	**−1.52**	+16.64
2K context	**100.00**	0.00	28.00	**−2.64**	−20.24	−7.60	+12.65
4K context	**89.00**	0.00	9.50	**−3.46**	−21.25	−15.29	+5.96
8K context	0.00	0.00	**3.50**	−29.02	−21.67	**−18.65**	+3.02
Phonebook (exact match)							
16 entries	**54.50**	0.00	0.00	**−1.81**	−23.16	−20.26	+2.90
32 entries	**52.00**	0.00	0.00	**−2.25**	−23.18	−21.97	+1.21
64 entries	**45.50**	0.00	0.00	**−3.62**	−23.12	−22.52	+0.60
128 entries	**24.00**	0.00	0.00	**−7.06**	−23.11	−22.60	+0.51
Based (case-insensitive contains accuracy)							
SWDE	**60.58**	6.57	17.46	**−8.62**	−15.95	−13.74	+2.21
FDA	**50.45**	0.36	9.53	**−12.10**	−51.60	−36.77	+14.83
SQuAD	11.33	14.91	**33.78**	−18.61	−18.79	**−15.06**	+3.73
NQ	**16.44**	7.95	11.43	**−9.75**	−17.11	−15.27	+1.85
TriviaQA	**56.93**	42.83	54.27	**−9.54**	−12.03	−9.87	+2.15
DROP	**23.47**	14.04	21.31	**−5.54**	−8.44	−6.07	+2.37
6-task accuracy avg	**36.53**	14.45	24.63	—	—	—	—

D.8 GROUND-TRUTH LIKELIHOOD ON RECALL-HEAVY TASKS

Alongside generation accuracy, Table 15 reports the conditional log-likelihood of the reference answer under teacher forcing, using the same three 1.3 B scaling-study checkpoints without further training. For a prompt x and reference answer y, we score

$$\ell_\theta(x, y) = \sum_{t=1}^{|y|} \log p_\theta(y_t \mid x, y_{<t}), \tag{10}$$

including only answer tokens. We retain the task-specific prompt preprocessing and truncation used in the original evaluation. Identical tokenized prompts and references are used across models; for multiple-reference examples, the first dataset-ordered reference is fixed before comparing models. Scores include the task-defined answer delimiter and exclude EOS. We report the mean per-example log-likelihood in nats and the paired IA minus S4D-only difference. Because answer lengths and formats differ across tasks, we compare log-likelihoods only within each task.

IA has higher mean reference-answer log-likelihood than S4D-only in all 14 recall settings. On Phonebook, both models have zero exact-match accuracy in Table 15, but the likelihood difference is positive at every list size, ranging from 0.51 to 2.90 nats per answer.

E GENOMIC MODELING DETAILS

E.1 PRETRAINING

Table 16: HG38 masked language modeling at 1K context and 10,000 updates. Caduceus-IA is the final configuration of the width-256 incremental ablation. Native Caduceus is the width-256 Caduceus-PS baseline; its initialization and residual-precision settings are detailed in Appendix B.2.

Model	Parameters	Validation PPL ↓
Native Caduceus	1.935M	2.612
Caduceus-IA	1.936M	2.614

E.2 GENOMIC DOWNSTREAM RESULTS

Baseline results are cited directly from Schiff et al. (2024). Following their evaluation protocol, GenomicBenchmarks and Nucleotide Transformer report the mean ± range across split seeds (5 and 10 splits, respectively). For variant-effect prediction (VEP), Caduceus-IA reports the mean ± sample standard deviation across five SVM fits, alongside published baseline means.

For GenomicBenchmarks, we evaluate a width-118 Caduceus-IA configuration (16 heads of width 7, 470,254 parameters), selected by pretraining validation loss. For Nucleotide Transformer, we evaluate the width-256 final Caduceus-IA configuration.

Fine-tuning updates the full backbone with mean pooling and no head dropout, using seed 2222, weight decay 0.1, gradient clipping at 1.0, and BF16 mixed precision with TF32 disabled. GenomicBenchmarks uses 10 epochs and effective batch size 256. Nucleotide Transformer uses 20 epochs. Caduceus-IA-PS uses learning rate 0.001, except for the Nucleotide Transformer task enhancers_types, which uses 0.002; its Nucleotide Transformer effective batch size is 512 for H3K4me2, H3K4me3, enhancers, enhancers_types, promoter_tata and splice_sites_all, and 128 for the remaining 12 tasks. Caduceus-IA-Ph uses the task-specific learning rates and effective batch sizes reported for Caduceus-Ph by Schiff et al. (2024, Tables 5–6), with reverse-complement augmentation during fine-tuning following their Section 4.2; Caduceus-IA-PS is fine-tuned without reverse-complement augmentation. The learning-rate schedule uses a 1% warmup fraction and minimum learning-rate ratio 0.1. Runs use the prescribed dataset splits and a fixed number of epochs. For Nucleotide Transformer, we report the test score of the epoch with the best validation metric.

Variant-effect prediction. We evaluate causal-eQTL prediction using frozen reference and alternative sequence embeddings from the separate width-256, 7.7M-parameter Caduceus-IA model pretrained on HG38 at context length 131,072 for 50,000 updates (Section 5.2). The Caduceus-IA-PS and Caduceus-IA-Ph VEP models use chunk lengths of 64 and 16, respectively. An RBF-kernel SVM predicts variant effects from these embeddings. Variants are grouped by distance to the nearest transcription start site: 0–30 kb, 30–100 kb, and over 100 kb. For each group, we train five SVMs on independently sampled subsets of 5,000 training examples and evaluate on held-out chromosomes 9 and 10.

Caduceus-IA-PS matches Caduceus-PS below 30 kb, performs slightly better at 30–100 kb, and slightly worse beyond 100 kb (Table 19).

Table 17: GenomicBenchmarks accuracy. Mean ± full range (maximum minus minimum) over five split seeds. Published baselines are from Table 1 of Schiff et al. (2024); Caduceus-IA-PS and Caduceus-IA-Ph use the matched-scale width-118 configuration. Cad. denotes Caduceus and IA denotes Caduceus-IA. Each cell shows the mean above its range; bold indicates the highest mean in each row.

Task	CNN	HyenaDNA	Mamba	Cad. w/o equiv.	Cad. Ph	Cad. PS	IA Ph	IA PS
Mouse enhancers	0.715 ±0.087	0.780 ±0.025	0.743 ±0.054	0.770 ±0.058	0.754 ±0.074	**0.793 ±0.058**	0.751 ±0.050	0.775 ±0.066
Coding vs. intergenomic	0.892 ±0.008	0.904 ±0.005	0.904 ±0.004	0.908 ±0.003	0.915 ±0.003	0.910 ±0.003	**0.933 ±0.002**	0.931 ±0.005
Human vs. worm	0.942 ±0.002	0.964 ±0.002	0.967 ±0.002	0.970 ±0.003	**0.973 ±0.001**	0.968 ±0.002	**0.973 ±0.001**	0.969 ±0.002
Human enhancers Cohn	0.702 ±0.021	0.729 ±0.014	0.732 ±0.029	0.741 ±0.008	**0.747 ±0.004**	0.745 ±0.007	0.745 ±0.013	0.737 ±0.006
Human enhancer Ensembl	0.744 ±0.122	0.849 ±0.006	0.862 ±0.008	0.883 ±0.002	0.893 ±0.008	**0.900 ±0.006**	**0.900 ±0.002**	**0.900 ±0.009**
Human regulatory	0.872 ±0.005	0.869 ±0.012	0.814 ±0.211	0.871 ±0.007	0.872 ±0.011	0.873 ±0.007	**0.888 ±0.002**	0.881 ±0.002
Human OCR Ensembl	0.698 ±0.013	0.783 ±0.007	0.815 ±0.002	0.818 ±0.003	**0.828 ±0.006**	0.818 ±0.006	0.822 ±0.004	0.792 ±0.009
Human non-TATA promoters	0.861 ±0.009	0.944 ±0.002	0.933 ±0.007	0.933 ±0.006	0.946 ±0.007	0.945 ±0.010	**0.954 ±0.007**	0.950 ±0.004

Table 18: Nucleotide Transformer downstream tasks. Mean ± full range (maximum minus minimum) over ten split seeds. Published baselines are from Table 2 of Schiff et al. (2024). Cad. denotes Caduceus and IA denotes Caduceus-IA. Each cell shows the mean above its range; bold indicates the highest mean in each row.

Task	Metric	Enformer	DNABERT-2	NT-v2	HyenaDNA	Cad. Ph	Cad. PS	IA Ph	IA PS
H3	MCC	0.719 ±0.048	0.785 ±0.033	0.784 ±0.047	0.779 ±0.037	0.815 ±0.048	0.799 ±0.029	**0.827 ±0.032**	0.811 ±0.032
H3K14ac	MCC	0.288 ±0.077	0.516 ±0.028	0.551 ±0.021	0.612 ±0.065	0.631 ±0.026	0.541 ±0.212	**0.652 ±0.022**	0.625 ±0.051
H3K36me3	MCC	0.344 ±0.055	0.591 ±0.020	0.625 ±0.013	0.613 ±0.041	0.601 ±0.129	0.609 ±0.109	**0.658 ±0.024**	0.641 ±0.035
H3K4me1	MCC	0.291 ±0.061	0.511 ±0.028	**0.550 ±0.021**	0.512 ±0.024	0.523 ±0.039	0.488 ±0.102	0.546 ±0.032	0.509 ±0.038
H3K4me2	MCC	0.211 ±0.069	0.336 ±0.040	0.319 ±0.045	0.455 ±0.095	0.487 ±0.170	0.388 ±0.101	**0.520 ±0.059**	0.490 ±0.042
H3K4me3	MCC	0.158 ±0.072	0.352 ±0.077	0.410 ±0.033	0.549 ±0.056	0.544 ±0.045	0.440 ±0.202	**0.571 ±0.033**	0.521 ±0.037
H3K79me3	MCC	0.496 ±0.042	0.613 ±0.030	0.626 ±0.026	0.672 ±0.048	0.697 ±0.077	0.676 ±0.026	**0.716 ±0.015**	0.698 ±0.039
H3K9ac	MCC	0.420 ±0.063	0.542 ±0.029	0.562 ±0.040	0.581 ±0.061	0.622 ±0.030	0.604 ±0.048	**0.651 ±0.023**	0.607 ±0.041
H4	MCC	0.732 ±0.076	0.796 ±0.027	0.799 ±0.025	0.763 ±0.044	**0.811 ±0.022**	0.789 ±0.020	0.808 ±0.028	0.803 ±0.025
H4ac	MCC	0.273 ±0.063	0.463 ±0.041	0.495 ±0.032	0.564 ±0.038	0.621 ±0.054	0.525 ±0.240	**0.627 ±0.058**	0.593 ±0.038
Enhancers	MCC	0.451 ±0.108	0.516 ±0.098	**0.548 ±0.144**	0.517 ±0.117	0.546 ±0.073	0.491 ±0.066	0.513 ±0.050	0.522 ±0.071
Enhancer types	MCC	0.309 ±0.134	0.423 ±0.051	0.424 ±0.132	0.386 ±0.185	**0.439 ±0.054**	0.416 ±0.095	0.428 ±0.069	0.395 ±0.115
Promoter all	F1	0.954 ±0.006	0.971 ±0.006	**0.976 ±0.006**	0.960 ±0.005	0.970 ±0.004	0.967 ±0.004	0.968 ±0.003	0.965 ±0.006
Promoter no TATA	F1	0.955 ±0.010	0.972 ±0.005	**0.976 ±0.005**	0.959 ±0.008	0.969 ±0.011	0.968 ±0.006	0.969 ±0.006	0.966 ±0.008
Promoter TATA	F1	0.960 ±0.023	0.955 ±0.021	**0.966 ±0.013**	0.944 ±0.040	0.953 ±0.016	0.957 ±0.015	0.952 ±0.021	0.948 ±0.021
Splice sites all	Acc.	0.848 ±0.019	0.939 ±0.009	**0.983 ±0.008**	0.956 ±0.011	0.940 ±0.027	0.927 ±0.021	0.970 ±0.005	0.967 ±0.008
Splice acceptor	F1	0.914 ±0.028	0.975 ±0.006	**0.981 ±0.011**	0.958 ±0.010	0.937 ±0.033	0.936 ±0.077	0.973 ±0.007	0.967 ±0.010
Splice donor	F1	0.906 ±0.027	0.963 ±0.006	**0.985 ±0.022**	0.949 ±0.024	0.948 ±0.025	0.874 ±0.289	0.976 ±0.009	0.968 ±0.017

Table 19: Causal-eQTL variant-effect prediction. Test AUROC by distance to the nearest transcription start site. Caduceus-IA results are mean ± sample standard deviation over five SVM fits. Baseline means are taken from Schiff et al. (2024), Figure 4. Cad. denotes Caduceus and IA denotes Caduceus-IA. IA cells show the mean above its standard deviation; bold marks the highest mean in each row.

Distance	Enformer	NT-v2	HyenaDNA	Cad. w/o equiv.	Cad. Ph	Cad. PS	IA Ph	IA PS
0–30 kb	0.668	0.671	0.670	0.674	0.678	**0.679**	0.673 ±0.003	**0.679 ±0.005**
30–100 kb	**0.659**	0.593	0.599	0.623	0.648	0.612	0.639 ±0.018	0.620 ±0.018
$>$ 100 kb	0.563	0.540	0.569	0.562	0.580	**0.586**	0.562 ±0.010	0.574 ±0.009

F INFERENCE SCALING

F.1 AUTOREGRESSIVE DECODE: PREFIX-LENGTH SCALING

We benchmark the 1.3B architecture in Appendix A.3 with random weights on an RTX 6000 Ada (48 GB), using BF16 autocast, PyTorch 2.10, CUDA 12.8, and no `torch.compile`. We compare eager Softmax with the Flash SDPA backend against eager and CUDA-graphed IA. Both IA paths use chunked prefill with chunk size 2048 and query-first decode, which contracts the query before the SSM output projection without materializing key/value coefficient tensors. Softmax preallocates a KV cache for $N + 64$ tokens and attends to the growing valid prefix; it is not CUDA-graph captured.

Protocol. Each condition runs in a separate process. After warmup (two eight-step prefill/decode runs for eager paths; eight replays for graphs), we repeat prefill plus 64 CUDA-event-timed decode steps for 20 iterations (15 at B=32, 10 at B=64). Latency is the pooled median over steps 2–64; table ranges span these medians across completed prefixes. Decode speedup is the ratio of median per-prefix latencies (Softmax over graphed IA) over prefixes completed by both methods.

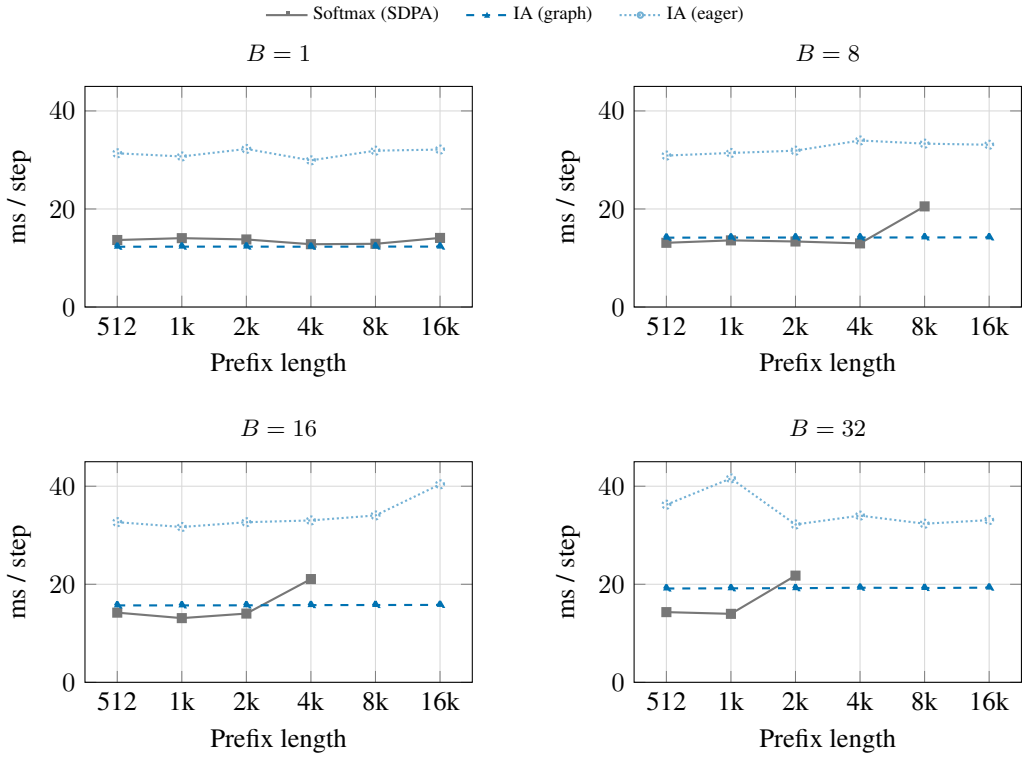


Figure 7: Steady-state decode latency versus prefix length for the 1.3 B model on a single RTX 6000 Ada (48 GB). Graphed IA decode remains approximately constant with prefix length. Softmax latency increases at long prefixes for $B \geq 8$. Curves stop at the largest prefix length completed by each method; Table 20 reports these limits.

F.2 PREFILL MEMORY SCALING

The prefill benchmark and the decode benchmark use different execution paths, so their out-of-memory limits differ. For example, Softmax prefill at B=8 fits at N=16,384 (Table 21), whereas Softmax decode at B=8 completes only up to N=8,192 (Table 20).

Table 20: Steady-state decode latency at 1.3 B on RTX 6000 Ada. "Range" is the minimum and maximum over completed prefix lengths, and "Largest N" is the largest successfully measured prefix in the tested grid. Speedup is defined in Appendix F.1.

	Softmax (SDPA)		IA			
B	Range (ms)	Largest N	Eager (ms)	Graph (ms)	Largest N	Speedup
1	12.81–14.09	16,384	29.92–32.25	12.30–12.33	16,384	1.11×
8	12.98–20.53	8,192	30.89–33.97	14.15–14.20	16,384	0.94×
16	13.10–21.06	4,096	31.68–40.38	15.69–15.79	16,384	0.90×
32	13.97–21.76	2,048	32.19–41.60	19.13–19.31	16,384	0.75×
64	24.10–29.06	1,024	64.93–66.64	38.41	1,024	0.69×

Table 21: Peak VRAM (GB) during prefill at 1.3B on a single RTX 6000 Ada (48 GB), bfloat16 mixed precision (`torch.autocast`). "Softmax (SDPA)" and "IA (non-chunked)" measure a full-sequence model forward, including vocabulary logits; "IA (chunked)" measures state construction with chunked IA prefill (chunk size 2048) and no vocabulary logits. "OOM" denotes out-of-memory. Bold indicates the lowest measured peak VRAM in each row.

B	N	Softmax (SDPA)	IA (non-chunked)	IA (chunked)
8	4,096	**14.65**	24.42	14.85
8	8,192	21.42	41.02	**15.39**
8	16,384	35.35	OOM	**12.38**
16	4,096	**21.21**	OOM	21.71
32	2,048	**21.16**	40.95	33.09
64	512	**14.59**	24.78	26.70

Table 22: Decode latency across model sizes on RTX 6000 Ada. Latencies are ranges over completed prefix lengths in $\{512, \ldots, 16{,}384\}$. Speedup is defined in Appendix F.1. For $B = 1$, all prefixes complete and the last column is the IA latency ratio at 16K versus 512 tokens. For $B = 8$, parenthesized lengths are the largest completed prefixes and the last column is the maximum-to-minimum Softmax latency ratio over its completed range.

Batch size $B = 1$				
Size	Softmax (ms)	IA graph (ms)	Speedup	IA 16K/512
125 M	6.47–6.91	2.37–2.38	2.81×	1.00×
350 M	12.89–14.14	5.51–5.53	2.46×	1.00×
760 M	13.13–15.07	8.45–8.48	1.63×	1.00×
1.3 B	12.81–14.09	12.30–12.33	1.11×	1.00×
Batch size $B = 8$				
Size	Softmax (ms)	IA graph (ms)	Speedup	Softmax max/min
125 M	6.80–7.48 (16 K)	2.81–2.83 (16 K)	2.46×	1.10×
350 M	13.38–16.97 (16 K)	6.54–6.56 (16 K)	2.10×	1.27×
760 M	14.58–15.68 (8 K)	9.86–9.92 (16 K)	1.53×	1.08×
1.3 B	12.98–20.53 (8 K)	14.15–14.20 (16 K)	0.94×	1.58×

F.3 DECODE LATENCY ACROSS MODEL SIZES

Graphed IA latency varies by less than 1% across completed prefix lengths at every model scale in both the $B{=}1$ and $B{=}8$ sweeps.

G TRAINING THROUGHPUT ON GH200

We compare full optimizer updates for the main-text 1.3B-scale language models and width-256, four-layer Caduceus-PS models, retaining the IA chunk length of 64. All models start from random initialization with seed 42.

Table 23: Training update cost on the same four GH200 GPUs. Throughput is the aggregate over all four GPUs; peak allocated memory is per GPU. Batch sizes are identical within each panel.

Model	Parameters	Median s/update	IQR (s)	Throughput (ktokens/s)	Peak memory (GiB/GPU)
Language modeling ($N = 4096$, 524,288 tokens/update)					
Softmax	1345.423M	3.598	0.005	145.7	48.77
S4D-only	1352.456M	5.285	0.017	99.2	65.50
IA	1352.407M	10.140	0.014	51.7	67.20
HG38 MLM ($N = 1024$, 1,048,576 input tokens/update)					
Native Caduceus-PS	1.935M	0.297	0.001	3534.4	10.40
Caduceus-IA-PS	1.936M	0.963	0.001	1088.4	58.48

Matched workloads. All five conditions run sequentially on the same node with four NVIDIA GH200 GPUs and bfloat16 mixed precision. Within each domain, GPU count, sequence length, microbatch size and gradient accumulation are identical. For language modeling, each GPU processes two sequences of length 4096 per microbatch and accumulates 16 microbatches, giving 524,288 tokens per optimizer update. For HG38 masked language modeling, each GPU processes 128 sequences of length 1024 and accumulates two microbatches, giving 1,024 sequences (1,048,576 input tokens) per update. We preload fixed windows from the corresponding FineWeb-Edu and HG38 training streams; inputs, targets and the 15% MLM masks are identical across models within each domain.

Implementation and precision. All language models use `torch.compile`; both genomic models use eager execution, matching their training configurations. TF32 is enabled for language modeling and disabled for genomics. Softmax uses PyTorch Flash SDPA, S4D-only uses its FFT convolution implementation, and the language-model IA uses Triton chunkwise kernels. Caduceus-IA uses TileLang chunkwise kernels; native Caduceus uses its Mamba implementation. Layer-level activation checkpointing is disabled. We preserve the genomic models' original residual precision: FP32 accumulation in Caduceus-IA and BF16 in native Caduceus. Each model uses the same fused AdamW parameter groups as in training, including the SSM learning-rate caps, with gradient clipping at 1.0.

Measurement. A timed update includes forward computation, task loss, backward computation, gradient accumulation, DDP gradient communication, clipping, and the optimizer update. Data loading, mask construction, validation, checkpoint writes and diagnostic logging are outside the timed interval. After five warmup updates, we synchronize CUDA at the boundaries of each of 20 measured updates and take the maximum duration across the four ranks. We report the median update time and its interquartile range (IQR); throughput is the global input-token count per update divided by the median duration. Peak memory is the maximum PyTorch allocation across ranks after resetting peak counters following warmup, and is reported per GPU. Losses and gradient norms were finite throughout, and no recompilation occurred during measurement. Each model was measured in a single run, so the IQR reflects update-to-update variability within that run.

The software environment is PyTorch 2.10.0 with CUDA 12.6, Triton 3.6.0, TileLang 0.1.14, Mamba-SSM 2.3.1, causal-conv1d 1.6.1, and NCCL 2.27.5.

At these matched settings, IA takes $2.82\times$ the update time of Softmax and $1.92\times$ that of S4D-only; Caduceus-IA-PS takes $3.25\times$ that of native Caduceus-PS.