Mamba: Linear-Time Sequence Modeling with Selective State Spaces

January 4, 2024

Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Albert Gu and Tri Dao

https://arxiv.org/abs/2312.00752

Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Albert Gu*[1] and Tri Dao*[2]

[1]Machine Learning Department, Carnegie Mellon University
[2]Department of Computer Science, Princeton University
agu@cs.cmu.edu, tri@tridao.me

Abstract

Foundation models, now powering most of the exciting applications in deep learning, are almost universally based on the Transformer architecture and its core attention module. Many subquadratic-time architectures such as linear attention, gated convolution and recurrent models, and structured state space models (SSMs) have been developed to address Transformers' computational inefficiency on long sequences, but they have not performed as well as attention on important modalities such as language. We identify that a key weakness of such models is their inability to perform content-based reasoning, and make several improvements. First, simply letting the SSM parameters be functions of the input addresses their weakness with discrete modalities, allowing the model to selectively propagate or forget information along the sequence length dimension depending on the current token. Second, even though this change prevents the use of efficient convolutions, we design a hardware-aware parallel algorithm in recurrent mode. We integrate these selective SSMs into a simplified end-to-end neural network architecture without attention or even MLP blocks (Mamba). Mamba enjoys fast inference (5× higher throughput than Transformers) and linear scaling in sequence length, and its performance improves on real data up to million-length sequences. As a general sequence model backbone, Mamba achieves state-of-the-art performance across several modalities such as language, audio, and genomics. On language modeling, our Mamba-3B model outperforms Transformers of the same size and matches Transformers twice its size, both in pretraining and downstream evaluation.

1 Introduction

Foundation models (FMs), or large models pretrained on massive data then adapted for downstream tasks, have emerged as an effective paradigm in modern machine learning. The backbone of these FMs are often sequence models, operating on arbitrary sequences of inputs from a wide variety of domains such as language, images, speech, audio, time series, and genomics (Brown et al. 2020; Dosovitskiy et al. 2020; Ismail Fawaz et al. 2019; Oord et al. 2016; Poli et al. 2023; Sutskever, Vinyals, and Quoc V Le 2014). While this concept is agnostic to a particular choice of model architecture, modern FMs are predominantly based on a single type of sequence model: the Transformer (Vaswani et al. 2017) and its core attention layer (Bahdanau, Cho, and Bengio 2015) The efficacy of self-attention is attributed to its ability to route information densely within a context window, allowing it to model complex data. However, this property brings fundamental drawbacks: an inability to model anything outside of a finite window, and quadratic scaling with respect to the window length. An enormous body of research has appeared on more efficient variants of attention to overcome these drawbacks (Tay, Dehghani, Bahri, et al. 2022), but often at the expense of the very properties that makes it effective. As of yet, none of these variants have been shown to be empirically effective at scale across domains.

Recently, structured state space sequence models (SSMs) (Gu, Goel, and Ré 2022; Gu, Johnson, Goel, et al. 2021) have emerged as a promising class of architectures for sequence modeling. These models can be interpreted as a combination of recurrent neural networks (RNNs) and convolutional neural networks (CNNs), with inspiration from classical state space models (Kalman 1960). This class of models can be computed very efficiently as either a recurrence or convolution, with linear or near-linear scaling in sequence length. Additionally, they have principled

*Equal contribution.

Mamba abstract

Foundation models, now powering most of the exciting applications in deep learning, are almost universally based on the Transformer architecture and its core attention module. Many subquadratic-time architectures such as linear attention, gated convolution and recurrent models, and structured state space models (SSMs) have been developed to address Transformers' computational inefficiency on long sequences, but they have not performed as well as attention on important modalities such as language. We identify that a key weakness of such models is their inability to perform content-based reasoning, and make several improvements. First, simply letting the SSM parameters be functions of the input addresses their weakness with discrete modalities, allowing the model to selectively propagate or forget information along the sequence length dimension depending on the current token. Second, even though this change prevents the use of efficient convolutions, we design a hardware-aware parallel algorithm in recurrent mode. We integrate these selective SSMs into a simplified end-to-end neural network architecture without attention or even MLP blocks (Mamba). Mamba enjoys fast inference (5× higher throughput than Transformers) and linear scaling in sequence length, and its performance improves on real data up to million-length sequences. As a general sequence model backbone, Mamba achieves state-of-the-art performance across several modalities such as language, audio, and genomics. On language modeling, our Mamba-3B model outperforms Transformers of the same size and matches Transformers twice its size, both in pretraining and downstream evaluation.

Mamba overview

- Builds on top of S4 state space model (SSM), also by Albert Gu
- *Selective state space model* allows some of the matrix trainable parameters in SSM to depend on the input
- The breaks fast training of S4 using convolutions, so they replace it with GPU hardware-optimized algorithm that does more computations than convolution but takes less time
- The new SSM, named S6, is placed in a block that also includes a MLP
- Moderate size experiments show ability to handle very long sequences such as 1 million, and show good scaling for language at 2.8B parameters
- People are anxious to see if a big Mamba with similar parameter count as a LLM will continue to scale to LLM level of performance

History of S4

- Long sequences are hard
  - RNNs have compact state but have difficulty past ~100 steps
  - Transformers have been best performance but have quadratic scaling
- Samuel Albanie video (https://youtu.be/ouF-H35atOY 0:00-6:10) mentions:
- Legendre Memory Units
- HiPPO: Recurrent Memory with Optimal Polynomial Projections
- Combining Recurrent, Convolutional and Continuous-time Models with Linear State-Space Layers
- Efficiently Modeling Long Sequences with Structured State Spaces (S4)

Linear State-Space Layer's 3 representations

The **Linear State-Space Layer (LSSL)** is a simple sequence model that maps a 1-dimensional function or sequence $u(t) \mapsto y(t)$ through an implicit state $x(t)$ by simulating a linear continuous-time state-space representation in discrete-time

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1}$$
$$y(t) = Cx(t) + Du(t), \tag{2}$$

Discretization. Then the discrete-time state-space model is

$$x_t = \overline{A}x_{t-1} + \overline{B}u_t \tag{4}$$
$$y_t = Cx_t + Du_t. \tag{5}$$

As a convolution. For simplicity let the initial state be $x_{-1} = 0$. Then (4)+(5) explicitly yields

$$y_k = C\left(\overline{A}\right)^k \overline{B}u_0 + C\left(\overline{A}\right)^{k-1} \overline{B}u_1 + \cdots + C\overline{AB}u_{k-1} + \overline{B}u_k + Du_k. \tag{6}$$

Then y is simply the (non-circular) convolution $y = \mathcal{K}_L(\overline{A}, \overline{B}, C) * u + Du$, where

$$\mathcal{K}_L(A, B, C) = \left(CA^iB\right)_{i\in[L]} \in \mathbb{R}^L = (CB, CAB, \ldots, CA^{L-1}B). \tag{7}$$

Pictorially, the 3 views of the LSSL

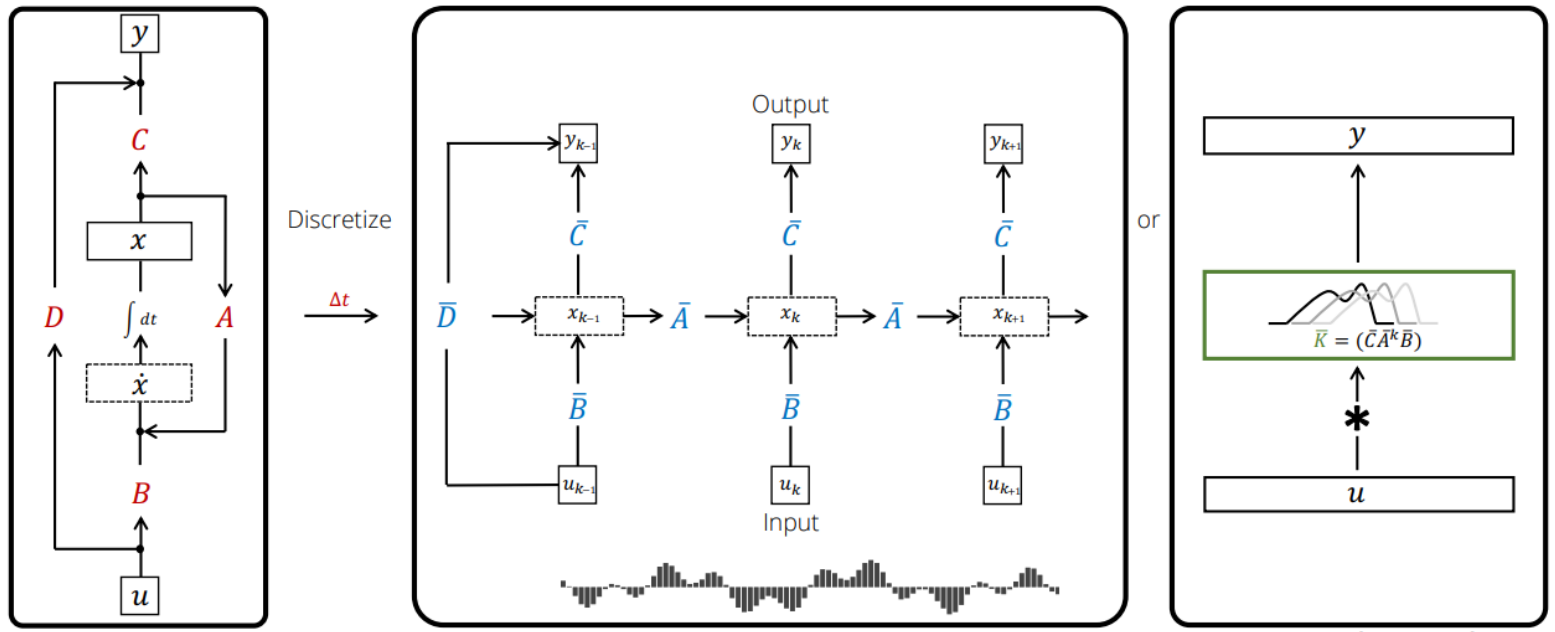

Figure 1: (**Three views of the LSSL**) A **Linear State Space Layer** layer is a map $u_t \in \mathbb{R} \to y_t \in \mathbb{R}$, where each feature $u_t \mapsto y_t$ is defined by discretizing a state-space model A, B, C, D with a parameter Δt. The underlying state space model defines a discrete recurrence through combining the state matrix A and timescale Δt into a transition matrix $\overline{A}$. (**Left**) As an implicit continuous model, irregularly-spaced data can be handled by discretizing the same matrix A using a different timescale Δt. (**Center**) As a recurrent model, inference can be performed efficiently by computing the layer *timewise* (i.e., one vertical slice at a time $(u_t, x_t, y_t), (u_{t+1}, x_{t+1}, y_{t+1}), \ldots$), by unrolling the linear recurrence. (**Right**) As a convolutional model, training can be performed efficiently by computing the layer *depthwise* in parallel (i.e., one horizontal slice at a time $(u_t)_{t\in[L]}, (y_t)_{t\in[L]}, \ldots$), by convolving with a particular filter.

More details about SSMs and S4

- Albert Gu video (https://www.youtube.com/live/EvQ3ncuriCM 0:00-27:00)
- Explains the 3 views of the state space model
- Challenges of deep SSMs
  - High powers of A matrix can lead to vanishing gradients
  - High powers of A matrix are expensive to compute
- S4 uses HiPPO operators for the A matrix
- Shows SOTA results on several long range tasks

How Mamba differs from S4

- Samuel Albanie video (https://youtu.be/ouF-H35atOY 6:10-16:00) adds:
- Motivation is to use selection as a means of compression
  - "Selection" refers to varying degree of attending to or ignoring different inputs
- Allows B,C,Δ matrix parameters to vary based on the input at each time step
  - But this breaks the conversion to convolutional kernel of S4
- Uses hardware-optimized algorithm that does fast recurrent calculation instead of using convolution for speed
  - Kernel fusion
  - Parallel scan seen in Simplified State Space Layers for Sequence Modeling (S5) paper
  - Recomputation when calculating gradients (avoids storage from forward pass)
- Combines architecture of prior SSM models with MLP block (like transformer)

Selective state space computation

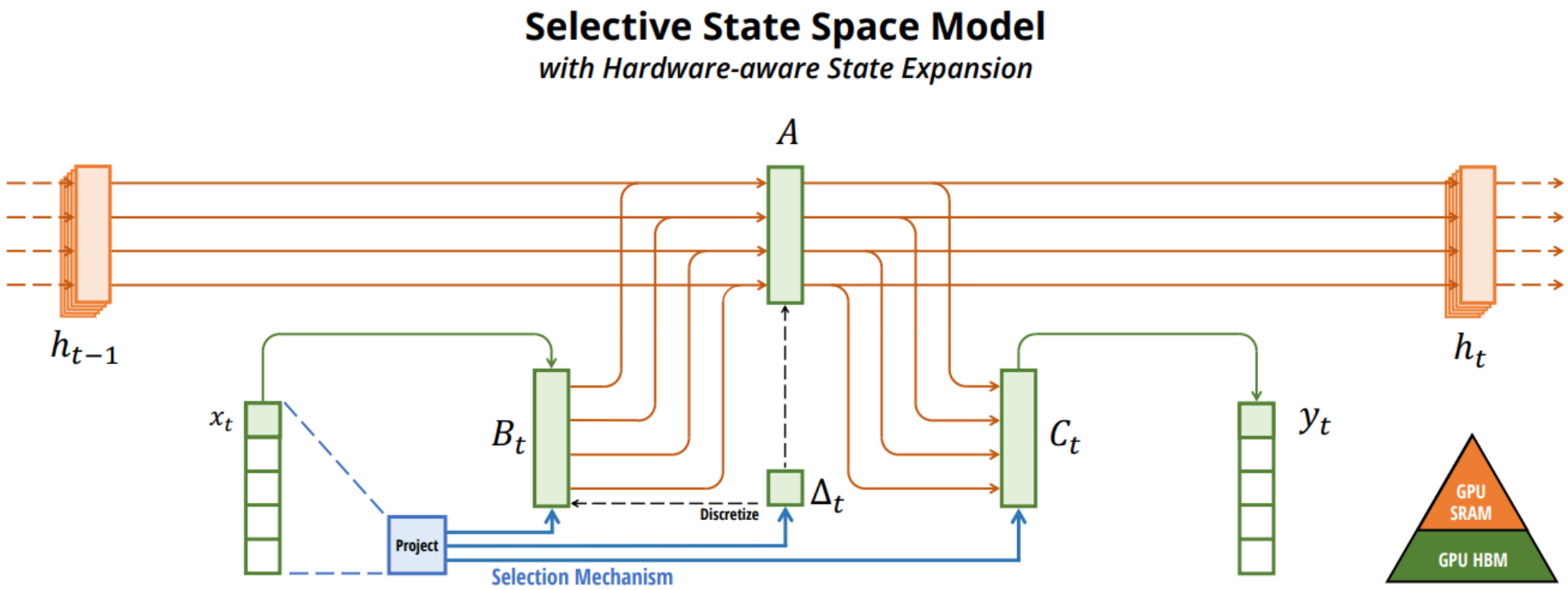

Figure 1: (**Overview**.) Structured SSMs independently map each channel (e.g. $D = 5$) of an input x to output y through a higher dimensional latent state h (e.g. $N = 4$). Prior SSMs avoid materializing this large effective state (DN, times batch size B and sequence length L) through clever alternate computation paths requiring time-invariance: the (Δ, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$) parameters are constant across time. Our selection mechanism adds back input-dependent dynamics, which also requires a careful hardware-aware algorithm to only materialize the expanded states in more efficient levels of the GPU memory hierarchy.

Mamba blocks

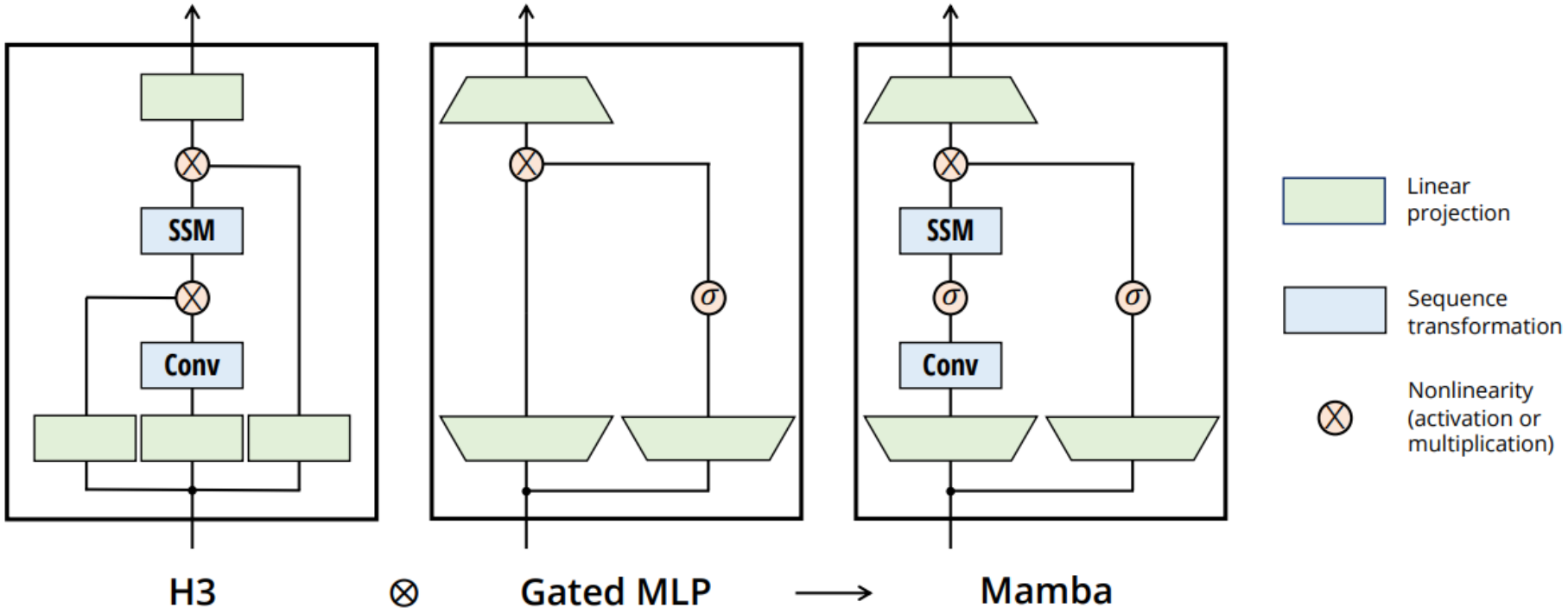

Figure 3: (**Architecture**.) Our simplified block design combines the H3 block, which is the basis of most SSM architectures, with the ubiquitous MLP block of modern neural networks. Instead of interleaving these two blocks, we simply repeat the Mamba block homogenously. Compared to the H3 block, Mamba replaces the first multiplicative gate with an activation function. Compared to the MLP block, Mamba adds an SSM to the main branch. For σ we use the SiLU / Swish activation (Hendrycks and Gimpel 2016; Ramachandran, Zoph, and Quoc V Le 2017).

Yannic's explanation of Mamba

- Yannic Kilcher video (https://youtu.be/9dSkvxS2EB0) explains differently:
- S4 has very fast inference because you can precompute coefficients of the convolutional kernel, then it's fast to apply it to your inputs at inference time
- Mamba relaxes the SSM constraint that all coefficients be constant, allowing some to vary based on the input at each timestep
- Yannic emphasizes that they still do not vary based on the hidden state
- GPU-optimized algorithm means it's fast even though it can't use convolutions

Results [1]

- Language modeling up to 1.3B parameters matches that of an attention transformer using a PaLM- and LLaMA-like training recipe

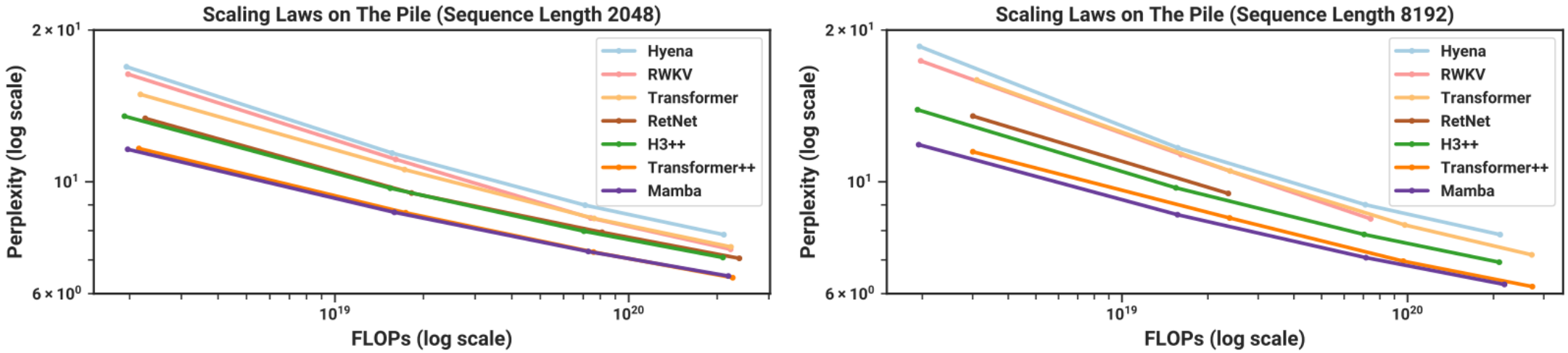

Figure 4: (**Scaling Laws.**) Models of size $\approx 125M$ to $\approx 1.3B$ parameters, trained on the Pile. Mamba scales better than all other attention-free models and is the first to match the performance of a very strong "Transformer++" recipe that has now become standard, particularly as the sequence length grows.

Results [2]

- Mamba models up to 2.8B parameters win against Pythia and other models with similar parameter counts on standard benchmarks

Table 3: (**Zero-shot Evaluations**.) Best results for each size in bold. We compare against open source LMs with various tokenizers, trained for up to 300B tokens. Pile refers to the validation split, comparing only against models trained on the same dataset and tokenizer (GPT-NeoX-20B). For each model size, Mamba is best-in-class on every single evaluation result, and generally matches baselines at twice the model size.

Model	Token.	Pile ppl ↓	LAMBADA ppl ↓	LAMBADA acc ↑	HellaSwag acc ↑	PIQA acc ↑	Arc-E acc ↑	Arc-C acc ↑	WinoGrande acc ↑	Average acc ↑
Hybrid H3-130M	GPT2	—	89.48	25.77	31.7	64.2	44.4	24.2	50.6	40.1
Pythia-160M	NeoX	29.64	38.10	33.0	30.2	61.4	43.2	24.1	**51.9**	40.6
Mamba-130M	NeoX	**10.56**	**16.07**	**44.3**	**35.3**	**64.5**	**48.0**	**24.3**	**51.9**	**44.7**
Hybrid H3-360M	GPT2	—	12.58	48.0	41.5	68.1	51.4	24.7	54.1	48.0
Pythia-410M	NeoX	9.95	10.84	51.4	40.6	66.9	52.1	24.6	53.8	48.2
Mamba-370M	NeoX	**8.28**	**8.14**	**55.6**	**46.5**	**69.5**	**55.1**	**28.0**	**55.3**	**50.0**
Pythia-1B	NeoX	7.82	7.92	56.1	47.2	70.7	57.0	27.1	53.5	51.9
Mamba-790M	NeoX	**7.33**	**6.02**	**62.7**	**55.1**	**72.1**	**61.2**	**29.5**	**56.1**	**57.1**
GPT-Neo 1.3B	GPT2	—	7.50	57.2	48.9	71.1	56.2	25.9	54.9	52.4
Hybrid H3-1.3B	GPT2	—	11.25	49.6	52.6	71.3	59.2	28.1	56.9	53.0
OPT-1.3B	OPT	—	6.64	58.0	53.7	72.4	56.7	29.6	59.5	55.0
Pythia-1.4B	NeoX	7.51	6.08	61.7	52.1	71.0	60.5	28.5	57.2	55.2
RWKV-1.5B	NeoX	7.70	7.04	56.4	52.5	72.4	60.5	29.4	54.6	54.3
Mamba-1.4B	NeoX	**6.80**	**5.04**	**64.9**	**59.1**	**74.2**	**65.5**	**32.8**	**61.5**	**59.7**
GPT-Neo 2.7B	GPT2	—	5.63	62.2	55.8	72.1	61.1	30.2	57.6	56.5
Hybrid H3-2.7B	GPT2	—	7.92	55.7	59.7	73.3	65.6	32.3	61.4	58.0
OPT-2.7B	OPT	—	5.12	63.6	60.6	74.8	60.8	31.3	61.0	58.7
Pythia-2.8B	NeoX	6.73	5.04	64.7	59.3	74.0	64.1	32.9	59.7	59.1
RWKV-3B	NeoX	7.00	5.24	63.9	59.6	73.7	67.8	33.1	59.6	59.6
Mamba-2.8B	NeoX	**6.22**	**4.23**	**69.2**	**66.1**	**75.2**	**69.7**	**36.3**	**63.5**	**63.3**
GPT-J-6B	GPT2	–	4.10	68.3	66.3	75.4	67.0	36.6	64.1	63.0
OPT-6.7B	OPT	–	4.25	67.7	67.2	76.3	65.6	34.9	65.5	62.9
Pythia-6.9B	NeoX	6.51	4.45	67.1	64.0	75.2	67.3	35.5	61.3	61.7
RWKV-7.4B	NeoX	6.31	4.38	67.2	65.5	76.1	67.8	37.5	61.0	62.5

Results [3]

- Also tested Mamba on DNA and audio data
- Performed a number of architecture ablations

Table 7: (**Ablations: Selective parameters.**) Δ is the most important parameter (Theorem 1), but using multiple selective parameters together synergizes.

Selective Δ	Selective B	Selective C	Perplexity
✗	✗	✗	10.93
✗	✓	✗	10.15
✗	✗	✓	9.98
✓	✗	✗	9.81
✓	✓	✓	8.71

Table 10: (**Ablations: SSM state dimension.**) (*Top*) Constant B and C (*Bottom*) Selective B and C. Increasing the SSM state dimension N, which can be viewed as an expansion factor on the dimension of the recurrent state, can significantly improve performance for a negligible cost in parameters/FLOPs, but only when B and C are also selective. Size of Δ projection fixed to 64.

State dimension N	Params (M)	Perplexity
1	367.1	9.88
2	367.4	9.86
4	368.0	9.82
8	369.1	9.82
16	371.5	9.81
1	367.1	9.73
2	367.4	9.40
4	368.0	9.09
8	369.1	8.84
16	371.5	8.71

Mamba conclusion

- State space models (specifically S4) are extended so that some parameters depend on the inputs, intuitively allowing model to decide how much to pay attention to or ignore certain inputs
- This change breaks the convolution approach to training, but they mitigate by developing a GPU algorithm that is very fast, despite doing more compute than convolutions
- Mamba tweaks the block architecture containing S6 and an MLP
- Perplexity and NLP benchmarks up to 2.8B param Mamba meet and exceed transformer LLMs
- Excitement to see if bigger Mamba models continue to scale, and if other things that work with transformer LLMs (prompting, in-context learning, instruction tuning, RLHF, etc.) also work with Mamba

References

- Hungry Hungry Hippos: Towards Language Modeling with State Space Models
  Daniel Y. Fu et al. (2022)
  https://arxiv.org/abs/2212.14052