STATE SPACE MODELS

WELCOME TO THE AI JUNGLE:

HIPPOS, HYENAS & MAMBAS (OH MY!)

KAHLIA HOGG

01 Motivation

02 SSMs

03 Mamba

04 MoE Mamba

05 Q&A

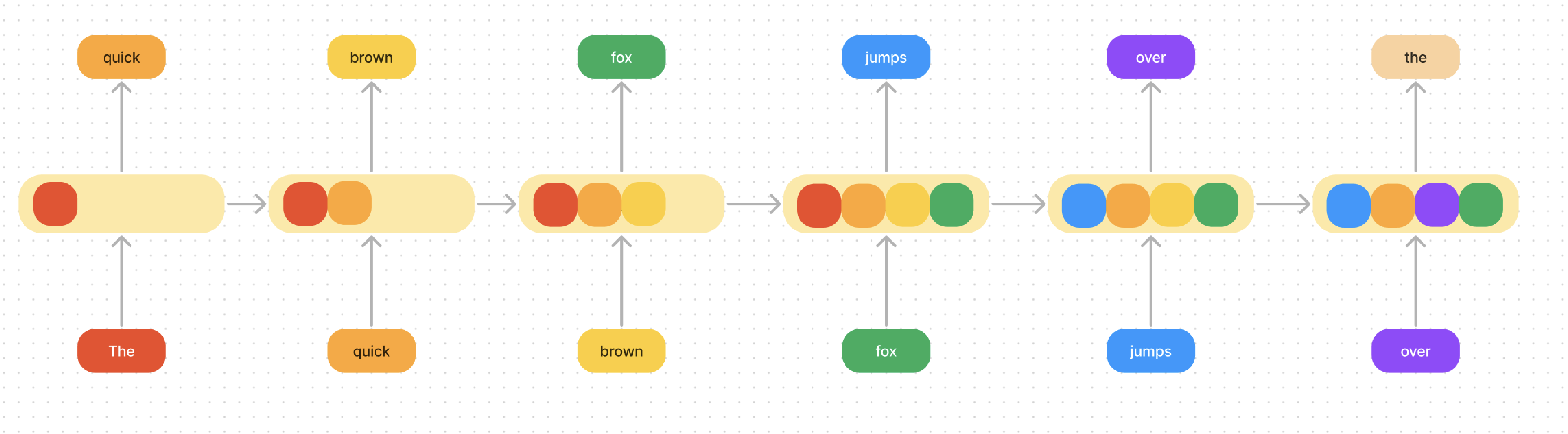
01 MOTIVATION

- Foundation models are emerging paradigm in AI/ML
- FMs = sequence models
- SOTA: transformer architecture & attention mechanism
- High expressive power but computationally inefficient
- Efficiency vs effectiveness tradeoff

In summary, the efficiency vs. effectiveness tradeoff of sequence models is characterized by how well they compress their state: efficient models must have a small state, while effective models must have a state that contains all necessary information from the context. In turn, we propose that a fundamental principle for building sequence

01 MOTIVATION

RNNs = efficient

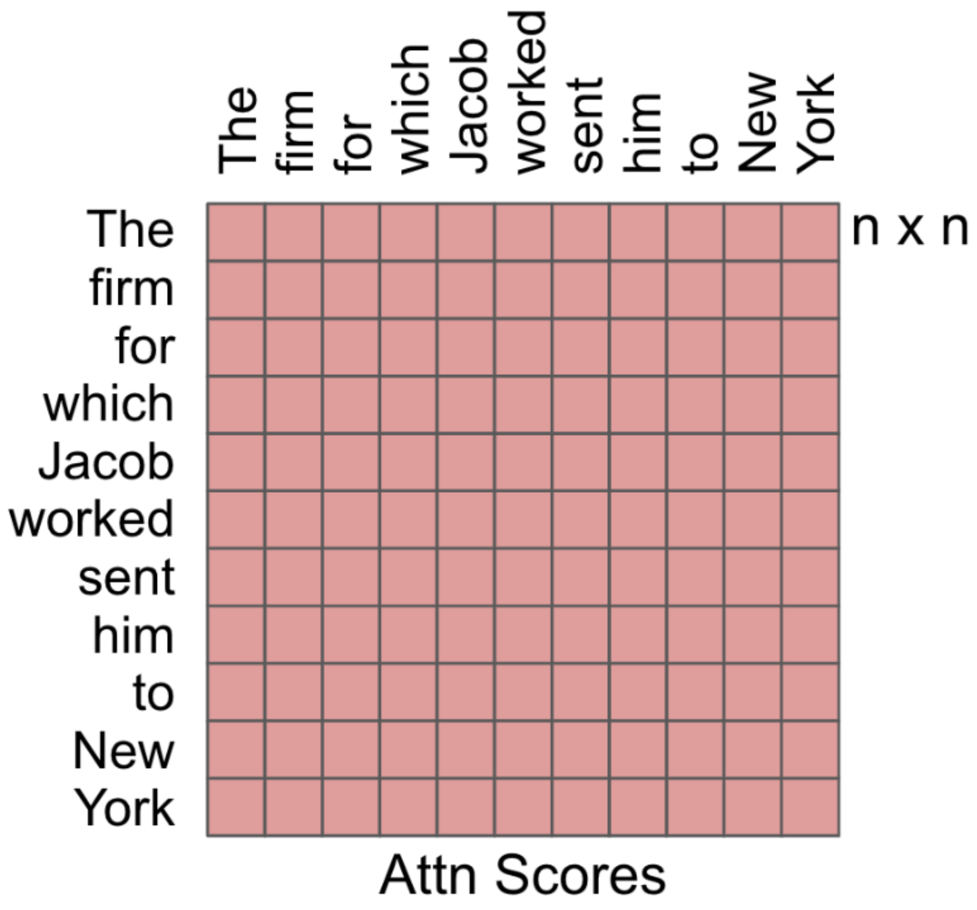


- ✅ Compressed (finite) state
- ✅ Fast O(1) inference
- ✅ O(N) training

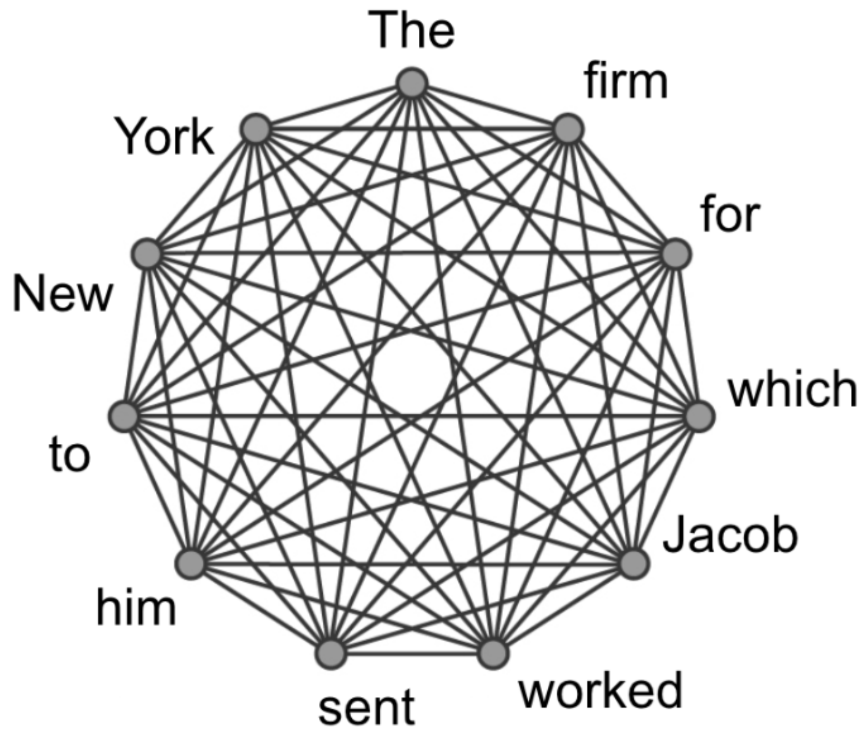
- ❌ "Forget" context
- ❌ Limited context window

01 MOTIVATION

Transformers = effective



=



- ✅ SOTA expressive power
- ✅ O(N) inference
- ✅ Parallelizable

- ❌ $O(N^2)$ training
- ❌ Limited context window

02 SSMs

State Space Models (S2)

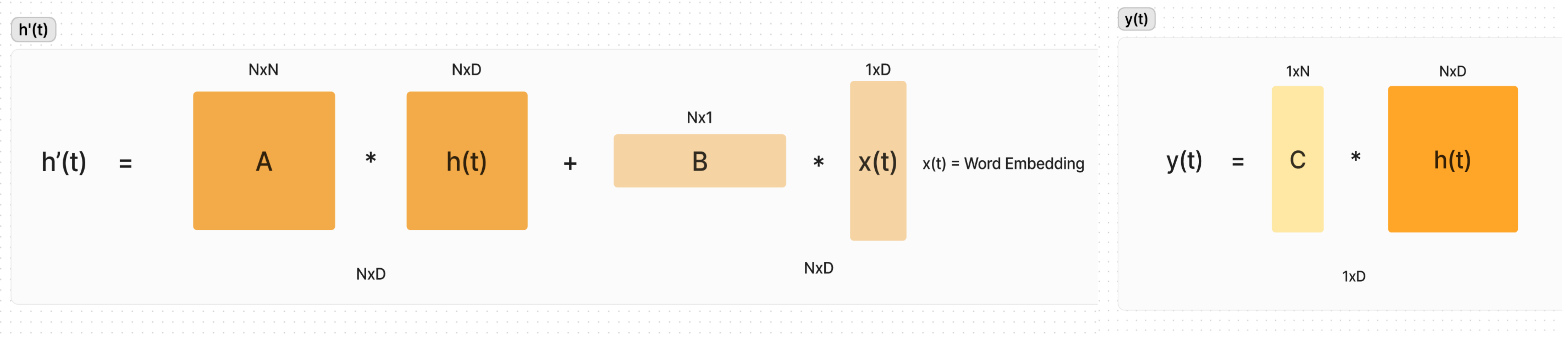
- Generalizes the idea of recurrent process + latent state

  → Kalman Filters, MDPs, DCMs, HMMs, RNNs, CNNs

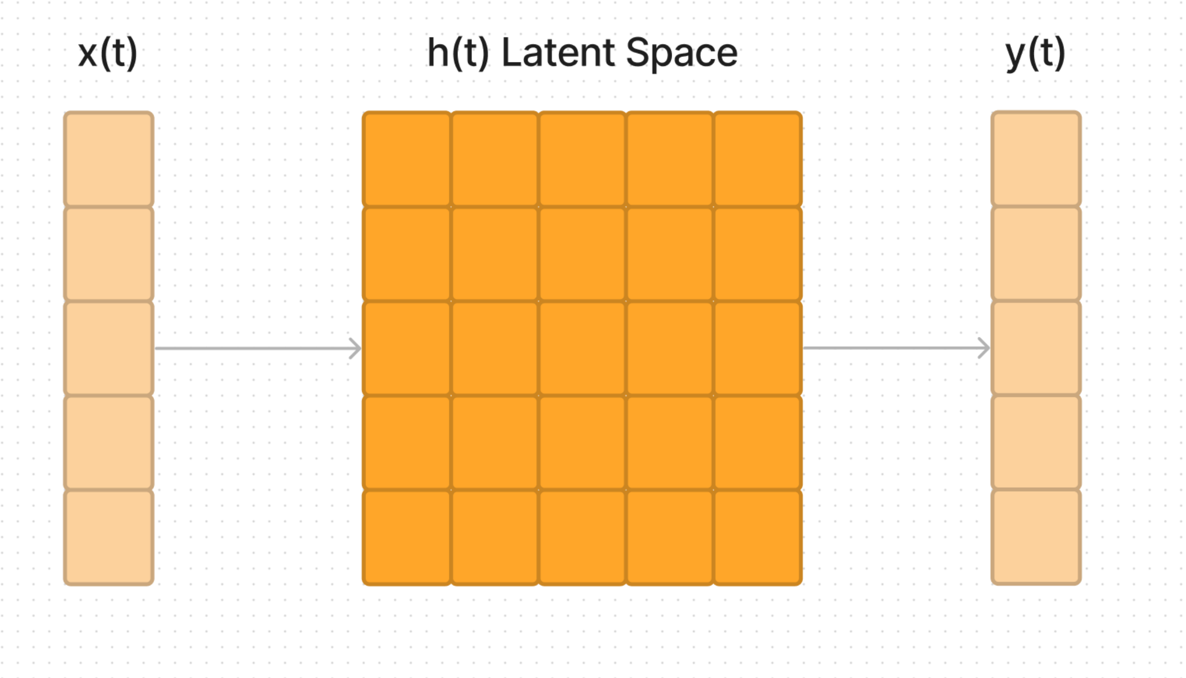
Structured State Space Sequence Models (S4)

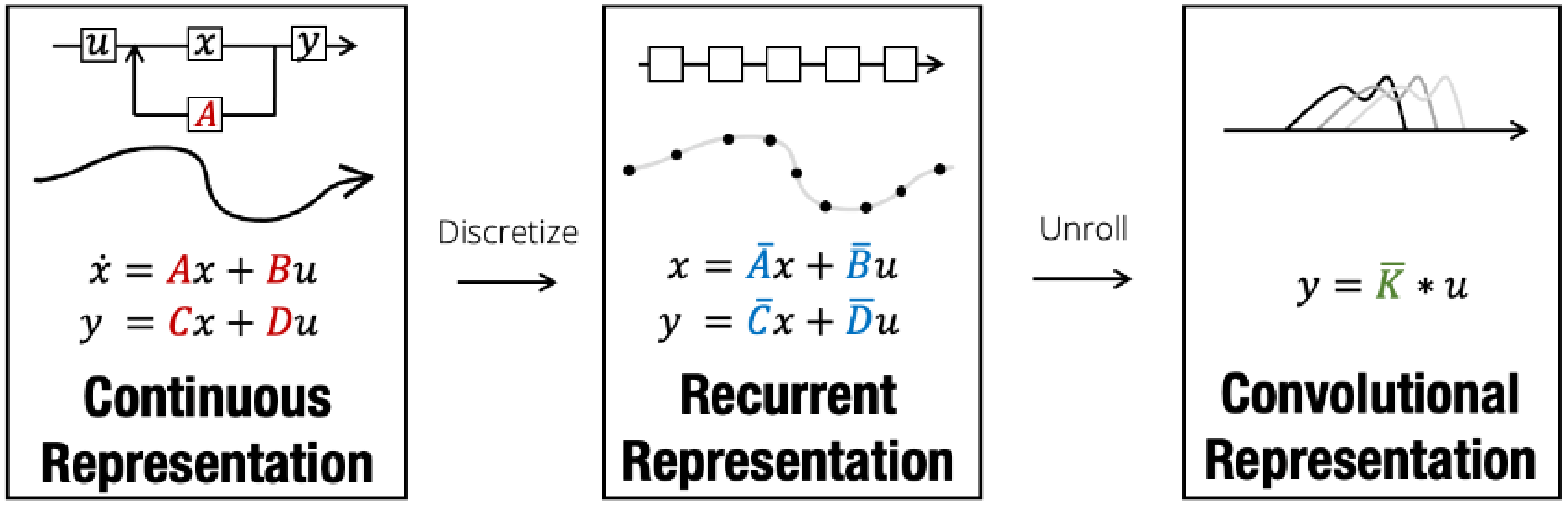
- Linear parameterization of S2 state equations → N-dim projection

$$h'(t) = Ah(t) + Bx(t)$$
$$y(t) = Ch(t)$$



02 SSMs





- Discretization step: (Δ,A,B,C)→(***A***,***B***,C)
- Dual forms:
    - “Recurrent” (discrete sequence) → efficient inference
    - “Convolutional” (continuous fxn) → parallelizable training
- “Unrolling” relies on Linear Time Invariance (LTI)
- Efficiently solve N-D latent space

02 SSMs

- S4 is a transformation that can be integrated into neural network architectures as SSM blocks/layers

Hungry Hungry Hippos
(HazyResearch, 2023)

Striped Hyena
(Together Research, 2023)

RWKV Eagle
(BlinkDL, 2023)

03 MAMBA

Goal: Expressive power of attention with near linear S4 efficiency

Selection Mechanism

- Allow SSM to focus on or filter out inputs
- Reparametrize Δ,A,B,C to be linear functions of the input
- L dim → Time varying → can't use efficient convolutional form

Algorithm 1 SSM (S4)

Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
$A : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
▷ Represents structured $N \times N$ matrix
$B : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$C : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\Delta : (\mathtt{D}) \leftarrow \tau_\Delta(\mathsf{Parameter})$
$\overline{A}, \overline{B} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{discretize}(\Delta, A, B)$
$y \leftarrow \mathsf{SSM}(\overline{A}, \overline{B}, C)(x)$
▷ Time-invariant: recurrence or convolution
return y

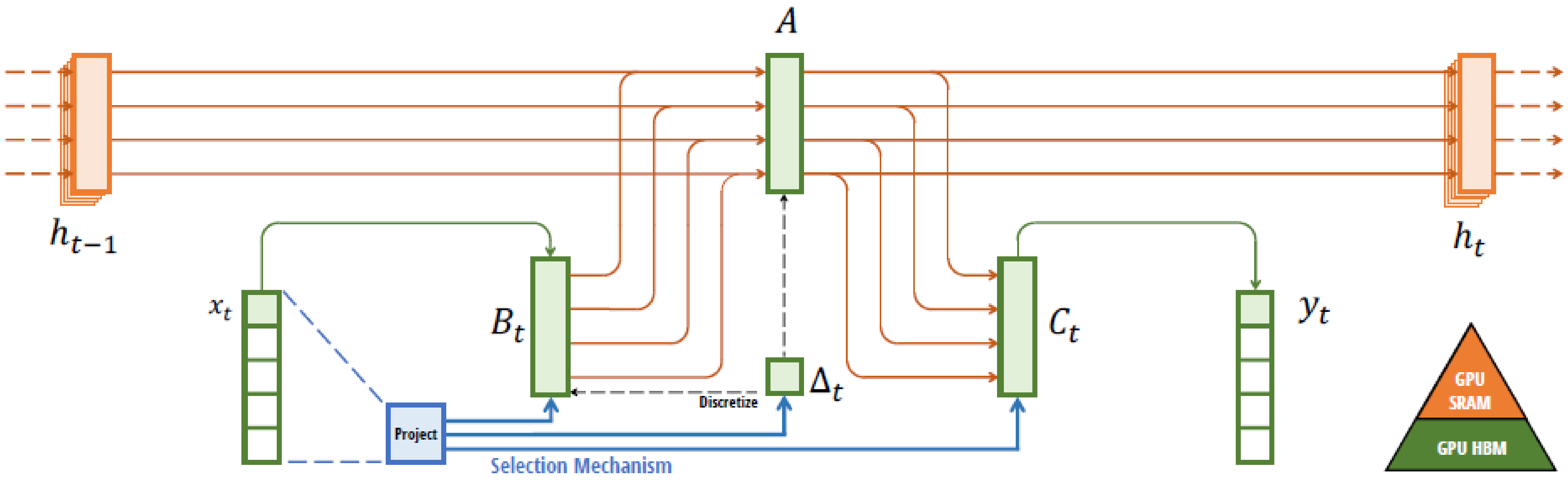
Algorithm 2 SSM + Selection (S6)

Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
$A : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
▷ Represents structured $N \times N$ matrix
$B : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_B(x)$
$C : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_C(x)$
$\Delta : (\mathtt{B}, \mathtt{L}, \mathtt{D}) \leftarrow \tau_\Delta(\mathsf{Parameter} + s_\Delta(x))$
$\overline{A}, \overline{B} : (\mathtt{B}, \mathtt{L}, \mathtt{D}, \mathtt{N}) \leftarrow \mathsf{discretize}(\Delta, A, B)$
$y \leftarrow \mathsf{SSM}(\overline{A}, \overline{B}, C)(x)$
▷ Time-varying: recurrence (*scan*) only
return y

03 MAMBA

Hardware Aware State Expansion

- Minimize memory IOs to maximize computation speed
- Materialize the state *h* in most efficient levels of GPU memory

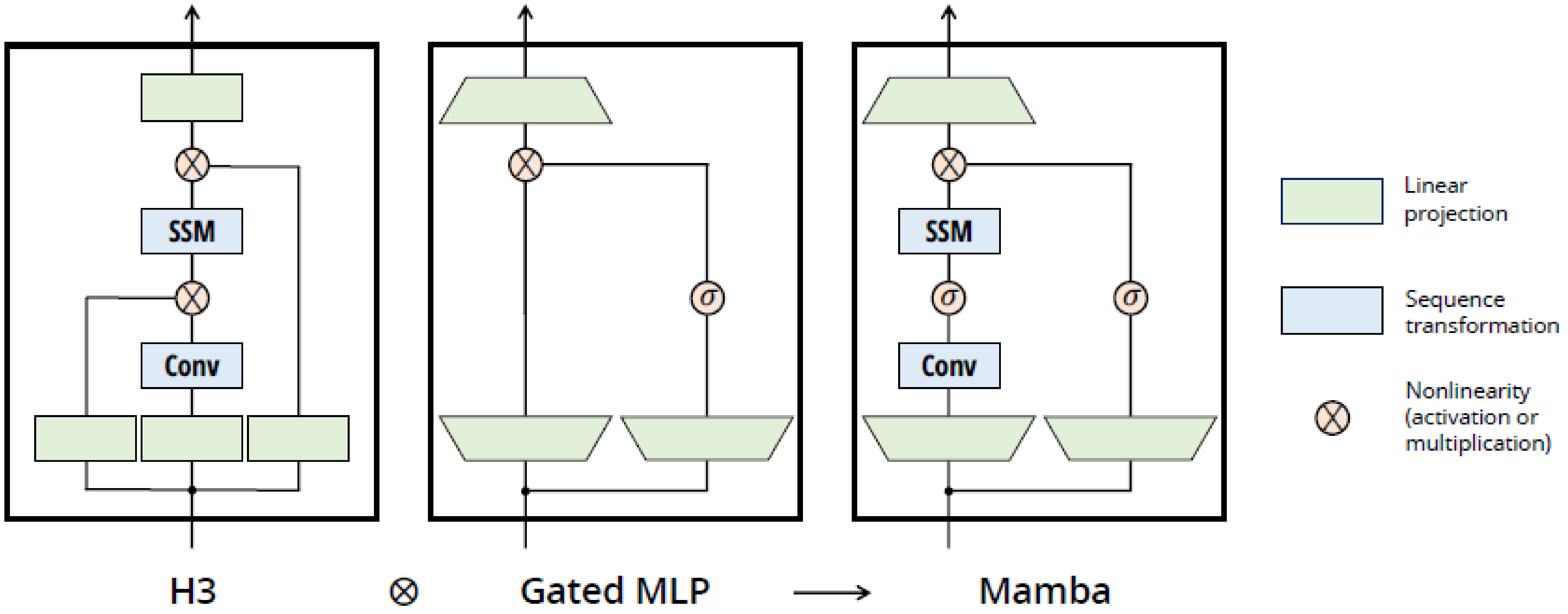


Concretely, instead of preparing the scan input $(\overline{A}, \overline{B})$ of size $(\mathtt{B}, \mathtt{L}, \mathtt{D}, \mathtt{N})$ in GPU HBM (high-bandwidth memory), we load the SSM parameters (Δ, A, B, C) directly from slow HBM to fast SRAM, perform the discretization and recurrence in SRAM, and then write the final outputs of size $(\mathtt{B}, \mathtt{L}, \mathtt{D})$ back to HBM.

03 MAMBA

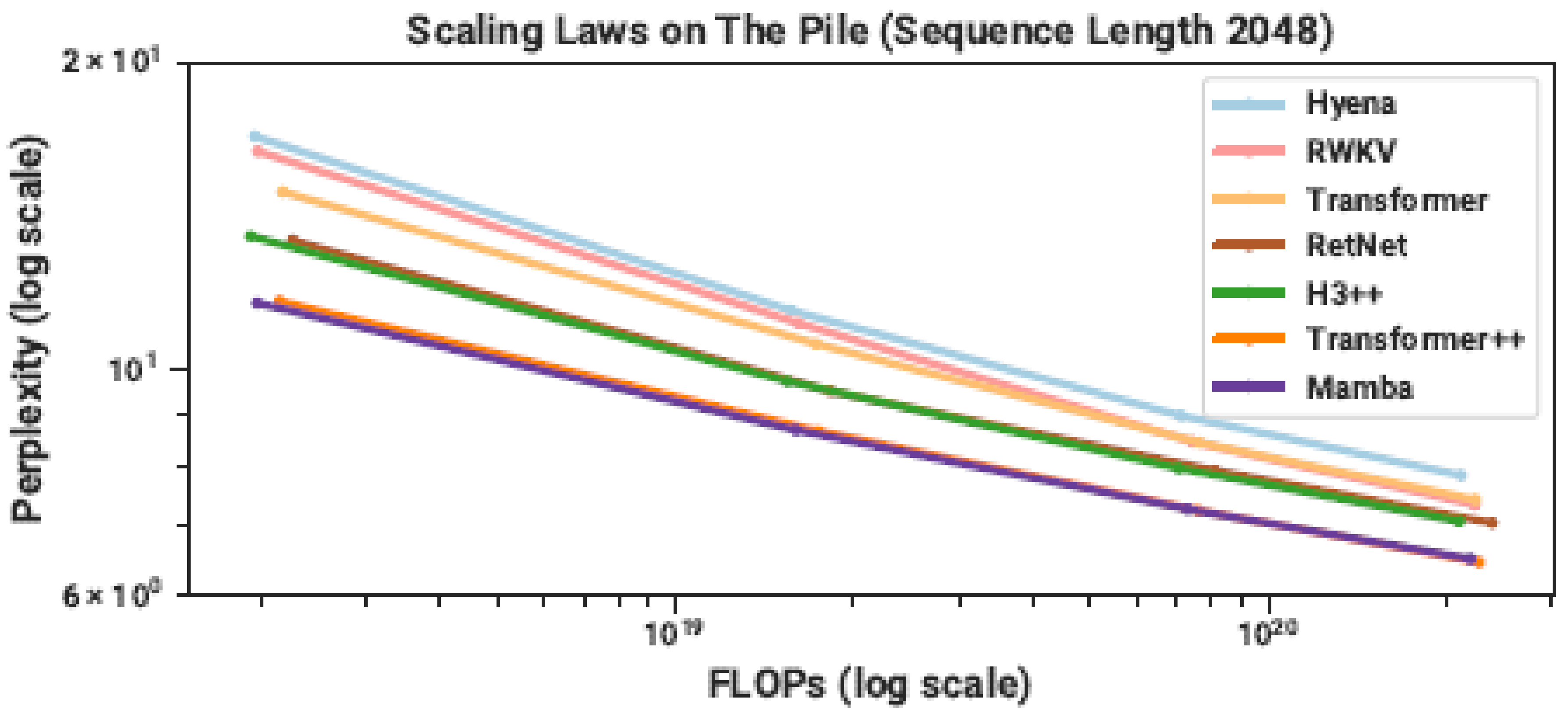
Structured State Space Sequence Models + Selective Scan (S6)

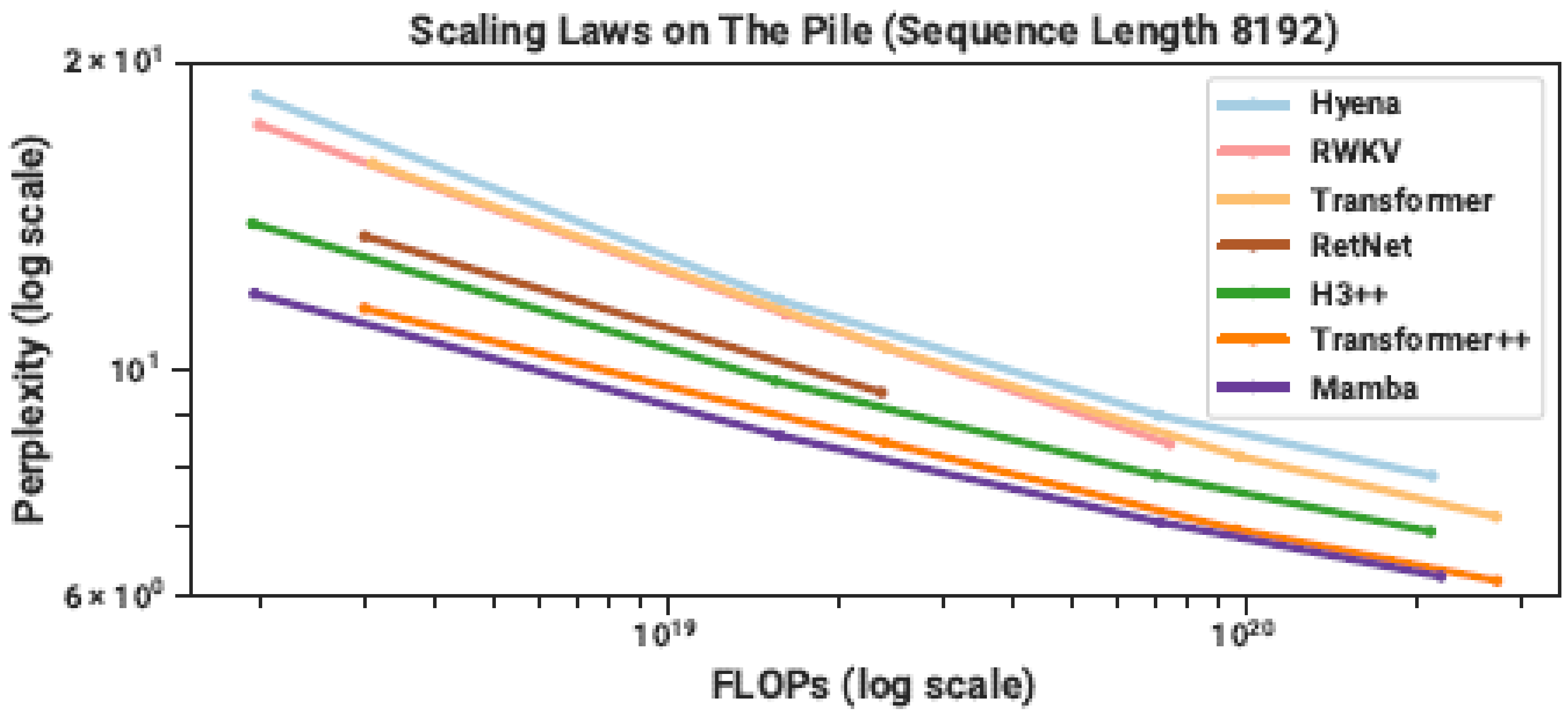
- Fast + Deadly + SSSSSS = Mamba 🐍



03 MAMBA

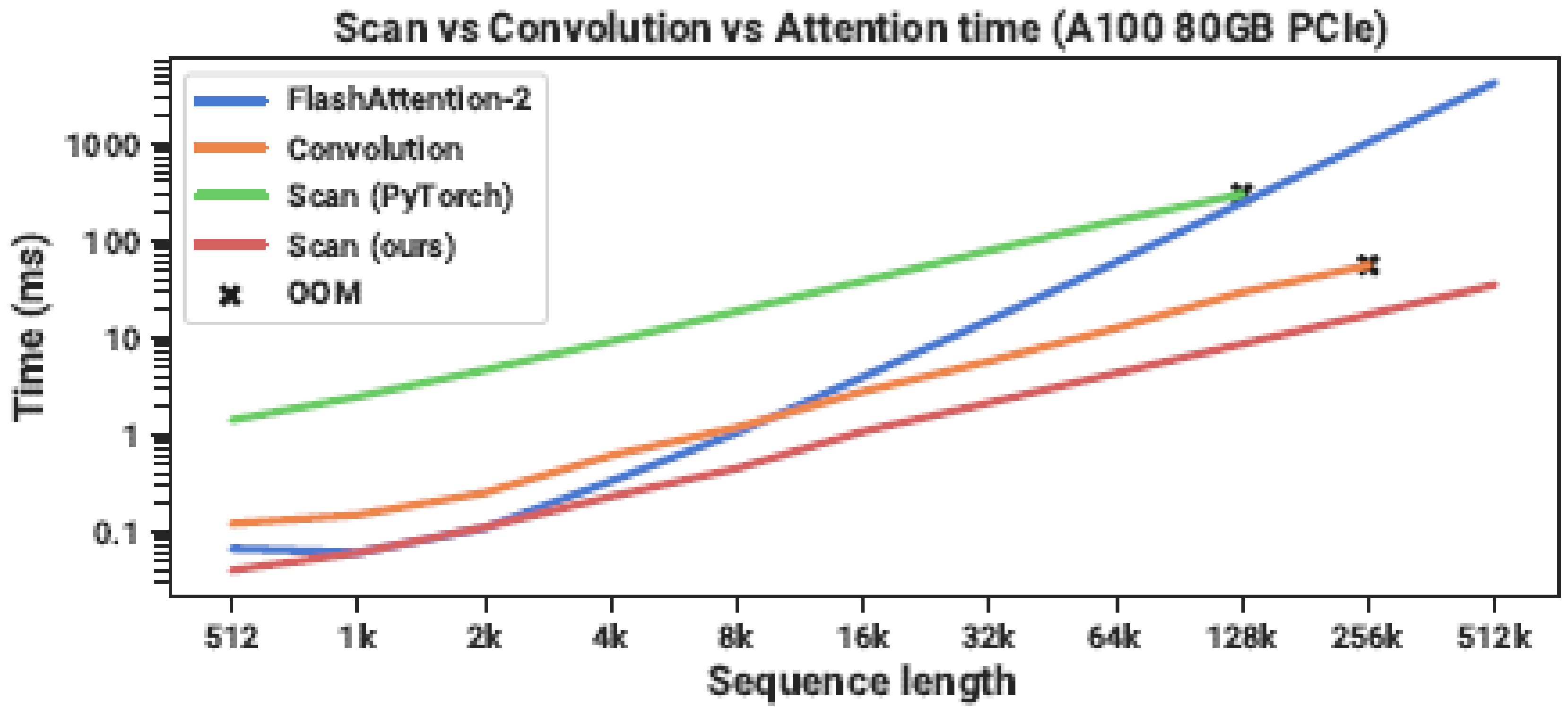
- Scaling laws outperform Transformer++ up to 1B parameters
- Chinchilla protocol: 20 training tokens/parameter





03 MAMBA

- “Unlimited” batch size since no KV cache



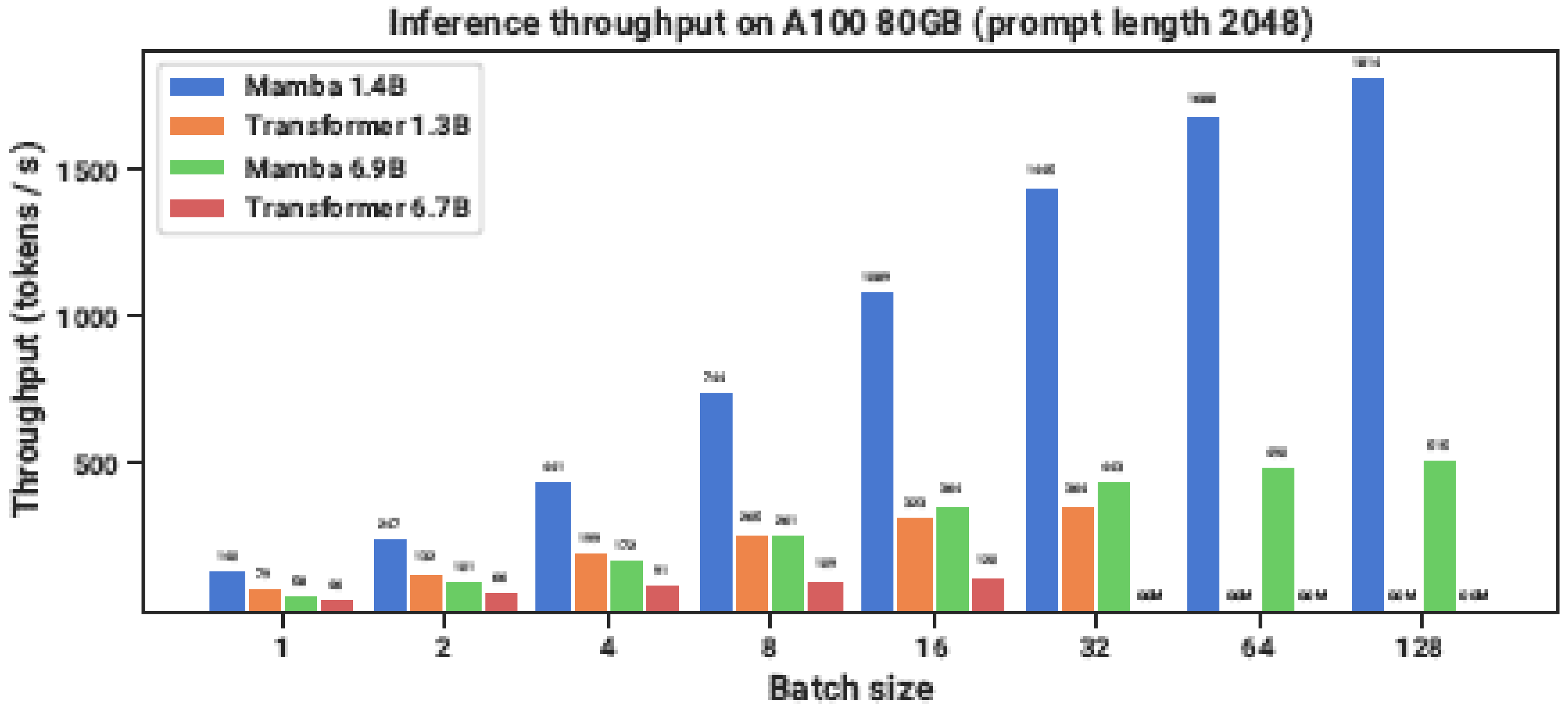


Figure 8: (**Efficiency Benchmarks.**) (*Left*) Training: our efficient scan is 40× faster than a standard implementation. (*Right*) Inference: as a recurrent model, Mamba can achieve 5× higher throughput than Transformers.

03 MAMBA

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

03 MAMBA

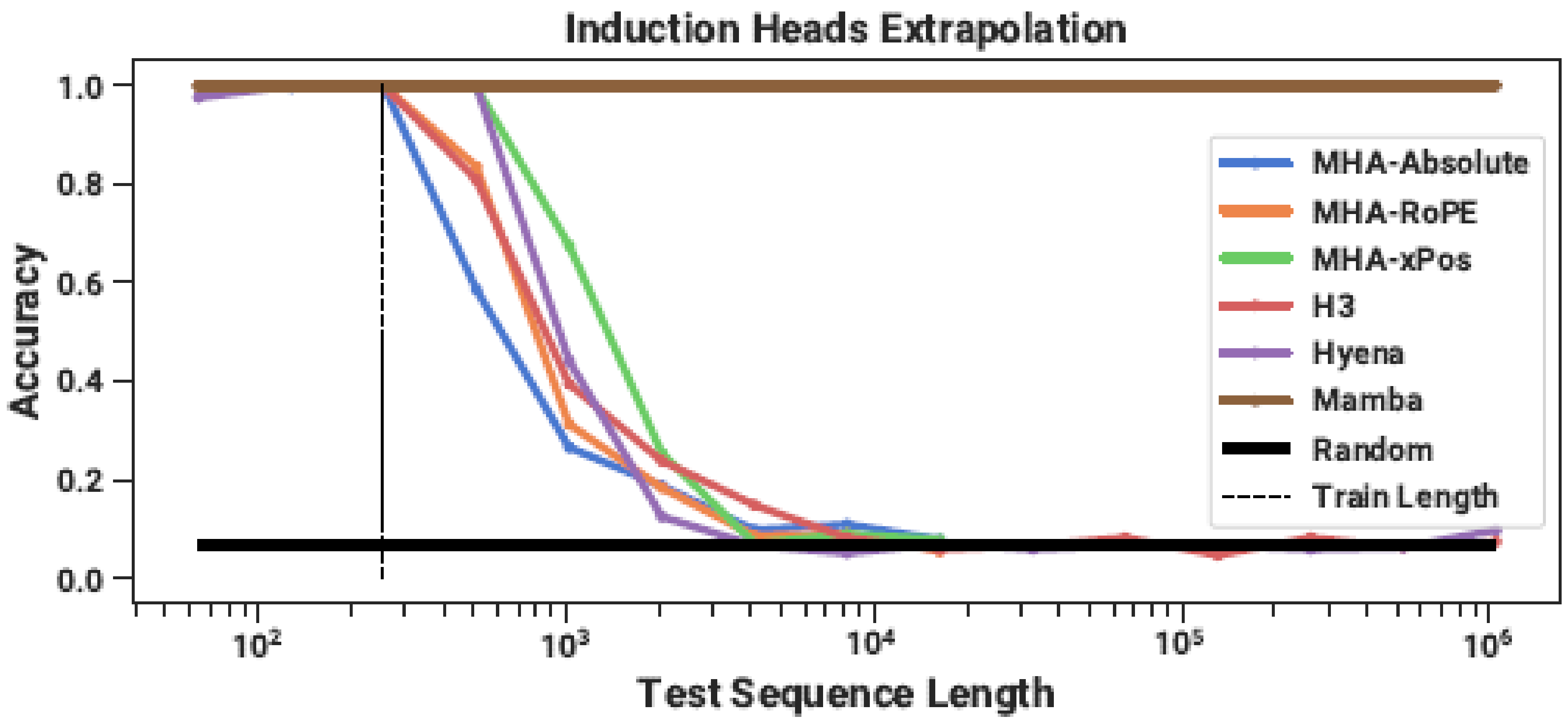


Table 11: (**Induction heads.**) Models are trained on sequence length $2^8 = 256$, and tested on various sequence lengths of $2^6 = 64$ up to $2^{20} = 1048576$. ✓ denotes perfect generalization accuracy, while ✗ denotes out of memory.

Model	Params	Test Accuracy (%) at Sequence Length														
		2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}
MHA-Abs	137K	✓	99.6	100.0	58.6	26.6	18.8	9.8	10.9	7.8	✗	✗	✗	✗	✗	✗
MHA-RoPE	137K	✓	✓	100.0	83.6	31.3	18.4	8.6	9.0	5.5	✗	✗	✗	✗	✗	✗
MHA-xPos	137K	✓	✓	100.0	99.6	67.6	25.4	7.0	9.0	7.8	✗	✗	✗	✗	✗	✗
H3	153K	✓	✓	100.0	80.9	39.5	23.8	14.8	8.2	5.9	6.6	8.2	4.7	8.2	6.3	7.4
Hyena	69M*	97.7	✓	100.0	✓	44.1	12.5	6.6	5.1	7.0	5.9	6.6	6.6	5.9	6.3	9.8
Mamba	74K	✓	✓	100.0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

* Most of the parameters are in learnable positional encodings.

03 MAMBA

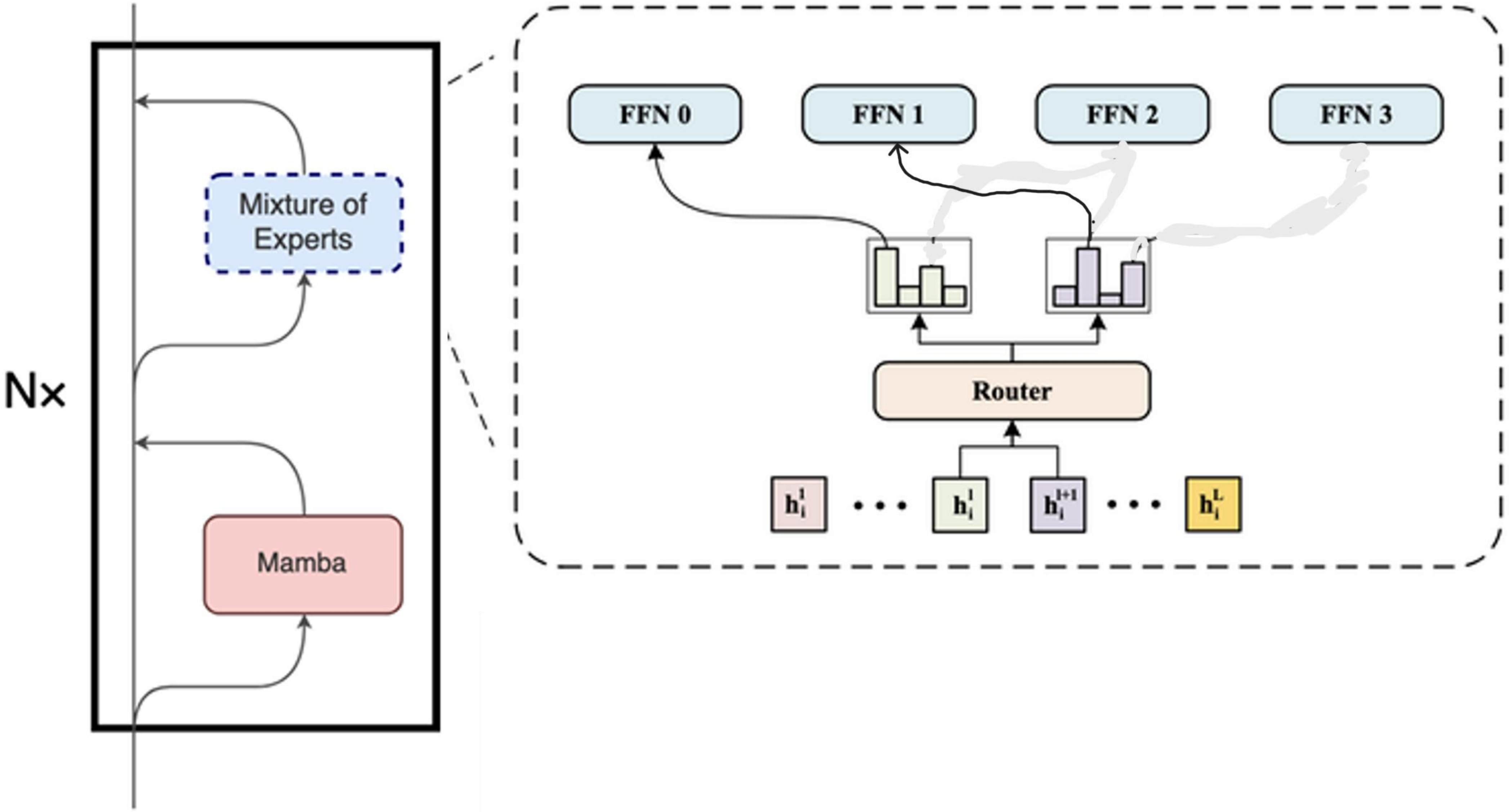
No Free Lunch

- “Linear” = O(BLND)
- Scaling: unknown empirical performance and engineering constraints beyond 2.8B parameters
- Downstream affordances: unknown fine-tuning, adaptation, prompting, in-context learning, instruction tuning, RLHF, quantization capability
- Continuous-Discrete spectrum: SSMs have a strong inductive bias toward continuous-time data modalities

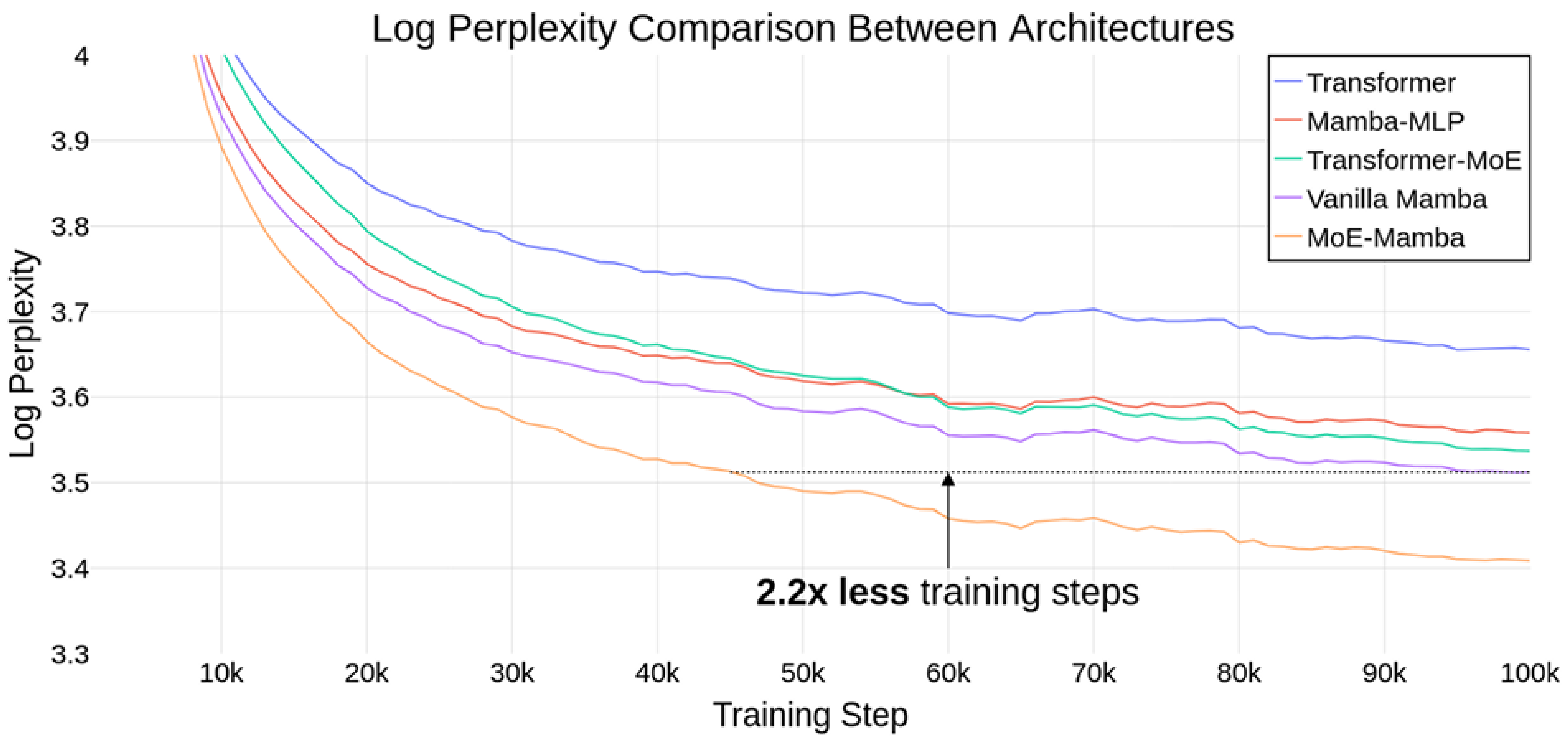
04 MoE-MAMBA

Mamba + Mixture of Experts

- Alternate each Mamba block with a MoE Switch block



04 MoE-MAMBA



Model	# Parameters	# Active Parameters per Token	Loss After 100k Steps	% Steps to Transformer Loss	% Steps to Vanilla Mamba Loss
Transformer	25M	25M	3.66	100%	>100%
Mamba-MLP	26M	26M	3.56	38%	>100%
Tranformer-MoE	545M	25M	3.54	42%	>100%
Vanilla Mamba	27M	27M	3.51	30%	100%
MoE-Mamba	416M	26M	**3.41**	**21%**	**46%**

References

Gu, A. & Dao, T., "Mamba: Linear-Time Sequence Modeling with Selective State Spaces", arXiv:2312.00752

Pioro et al., "MoE-Mamba: Efficient Selective State Space Models with Mixture of Experts", arXiv:2401.04081

Gu et al., "Structured State Spaces: Combining Continuous-Time, Recurrent, and Convolutional Models", Stanford HazyResearch, https://hazyresearch.stanford.edu/blog/2022-01-14-s4-3

Rush, S. & Karamcheti, S., "The Annotated S4", https://srush.github.io/annotated-s4/

Schoeninger, G., "Mamba: Linear-Time Sequence Modeling with Selective State Spaces - Arxiv Dives", Oxen AI, https://www.oxen.ai/blog/mamba-linear-time-sequence-modeling-with-selective-state-spaces-arxiv-dives

Google Research, "Constructing Transformers For Longer Sequences with Sparse Attention Methods", https://blog.research.google/2021/03/constructing-transformers-for-longer.html

Questions