Intro

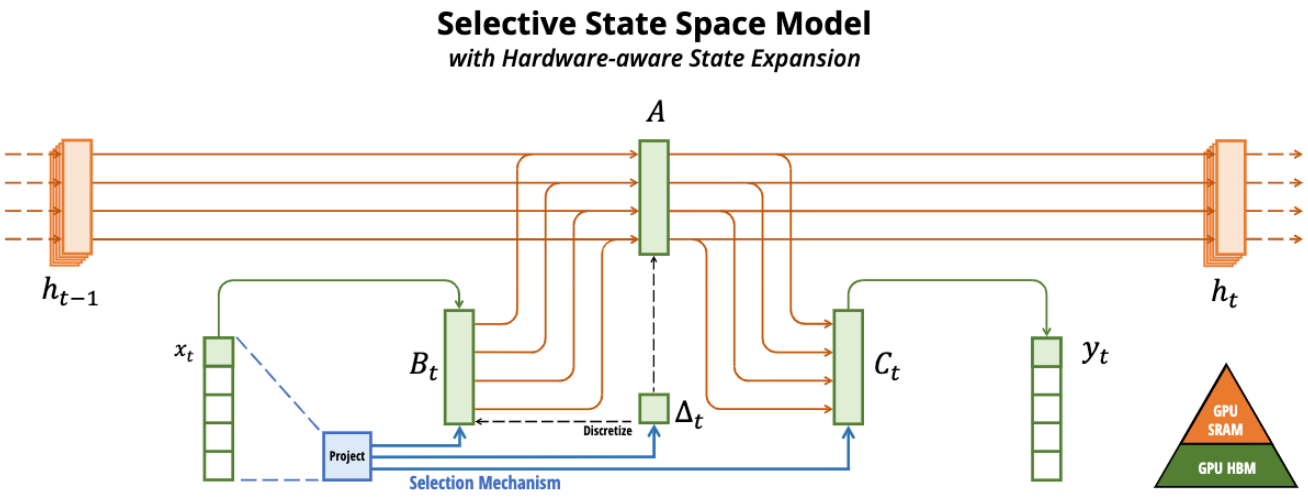


Figure 1: (**Overview.**) Structured SSMs independently map each channel (e.g. $D = 5$) of an input x to output y through a higher dimensional latent state h (e.g. $N = 4$). Prior SSMs avoid materializing this large effective state (DN, times batch size B and sequence length L) through clever alternate 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.

SSM: structured state space sequence models (SSMs): RNN + CNN 💡(state space models). niubi: computed efficiently, linear/non-linear scaling in seq length. 可以比较好解决long range dependency. 已经很火了在(auido | vision), 但是在建模discrete + information dense的数据上没那么effective(就是语言)

本文改进：

1. selection mechanism: efficiently select data in an input-dependent manner (attention?): parameterizing the SSM parameters based on the input.

2. Hardware-aware Alogithm: 利用好GPU(computes the model recurrently with a scan instead of convolution, but does not materialize the expanded state in order to avoid IO access between different levels of the GPU memory hierarchy??? 啥意思：避免GPU不同层级mem I/O?
    1. 效果： (up to 3× faster on A100 GPUs)
3. Architecture： SSM arch + FFN -> single block

改进之后达到:

1. High quality: 更好选择input data-> language和genomics上更强的效果
2. Fast training & inference: 自回归时候每一步不需要a cache of pre elements
3. Long context: up to 1M length

实验做在

1. Synthetics: copying and induction heads
2. Audio and Genomics: modeling audio waveforms and DNA sequences tasks (FID in challenging speech generation dataset)
3. Language Modeling: 5×generation throughput compared to Transformers of similar size; Mamba-3B's quality matches that of Transformers twice its size(3B甚至高于Pythia-7B)

State Space Models

S4: Structured state space sequence models:

- Like a continuous system maps a 1-d function $x(t) \in R \rightarrow y(t) \in R$ by $h(t) \in R$
    - 推导refer https://zhuanlan.zhihu.com/p/677787187

$$h'(t) = \boldsymbol{A}h(t) + \boldsymbol{B}x(t) \quad (1a) \qquad h_t = \overline{\boldsymbol{A}}h_{t-1} + \overline{\boldsymbol{B}}x_t \quad (2a) \qquad \overline{\boldsymbol{K}} = (C\overline{\boldsymbol{B}}, C\overline{\boldsymbol{AB}}, \dots, C\overline{\boldsymbol{A}}^{k}\overline{\boldsymbol{B}}, \dots) \quad (3a)$$

$$y(t) = \boldsymbol{C}h(t) \quad (1b) \qquad y_t = \boldsymbol{C}h_t \quad (2b) \qquad y = x * \overline{\boldsymbol{K}} \quad (3b)$$

Two stages:

1. *Discretization*
    1. $(\triangle, A, B, C) \rightarrow (\bar{A}, \bar{B}, C)$, transforms the "continuous parameters" (Δ, A, B) to "discrete parameters" $\bar{A}, \bar{B}$
    2. $\overline{A} = \exp(\Delta A) \quad \overline{B} = (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B$
    3. Why? resolution invariance, automatically ensuring that the model is properly normalized; 也有点像RNN的gating机制
2. *Computation*
    1. linear recurrence: 走上当年RNN的老路
    2. Global convolution: for efficient parallelizable training
- Linear Time Invariance: 算是缺陷?

    - (Δ, A, B, C), and consequently ($\bar{A}$, $\bar{B}$)as well, are fixed for all time-steps.
- Structure and Dimensions
    - $A \in \mathbb{R}^{N\times N}, B \in \mathbb{R}^{N\times 1}, C \in \mathbb{R}^{1\times N}$, can all be represented by N numbers
    - $x \in R^{BLD}$, the SSM is applied independently to each channel $d \in D$, the total hidden state has dimension DN for per x. Time and Memory的开销是$O(BLDN)$

SSSM (Selective State Space Models)

Motivation: Selection as a Means of Compression

Two tasks:

- Copying: 彩色序列中间有这Constant spacing: 不需要time-varing能力
    - Selective Copying: 彩色序列中间的spacing是random: 需要time-varing能力
- Induction Heads:
    - 用来解释ICL能力

这也是effeciency 和 effectivness的tradeoff:

- efficient: small mem state
- Effective: have a state that contains all necessary information from the context

因此解决的办法: selectiveness

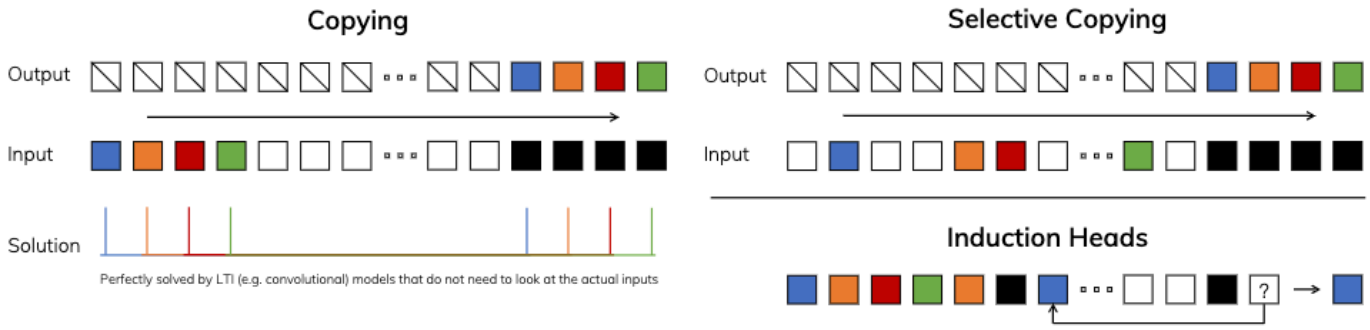


Figure 2: (*Left*) The standard version of the Copying task involves constant spacing between input and output elements and is easily solved by time-invariant models such as linear recurrences and global convolutions. (*Right Top*) The Selective Copying task has random spacing in between inputs and requires time-varying models that can *selectively* remember or ignore inputs depending on their content. (*Right Bottom*) The Induction Heads task is an example of associative recall that requires retrieving an answer based on context, a key ability for LLMs.

Improving SSMs with Selection

Algorithm 1 SSM (S4)

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

Algorithm 2 SSM + Selection (S6)

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

S6 : SSM + Selection

$s_B(x) = \text{Linear}_N(x), s_C(x) = \text{Linear}_N(x), s_\Delta(x) = \text{Broadcast}_D(\text{Linear}_1(x)), and确定\tau_\Delta = \text{soft}plus'$

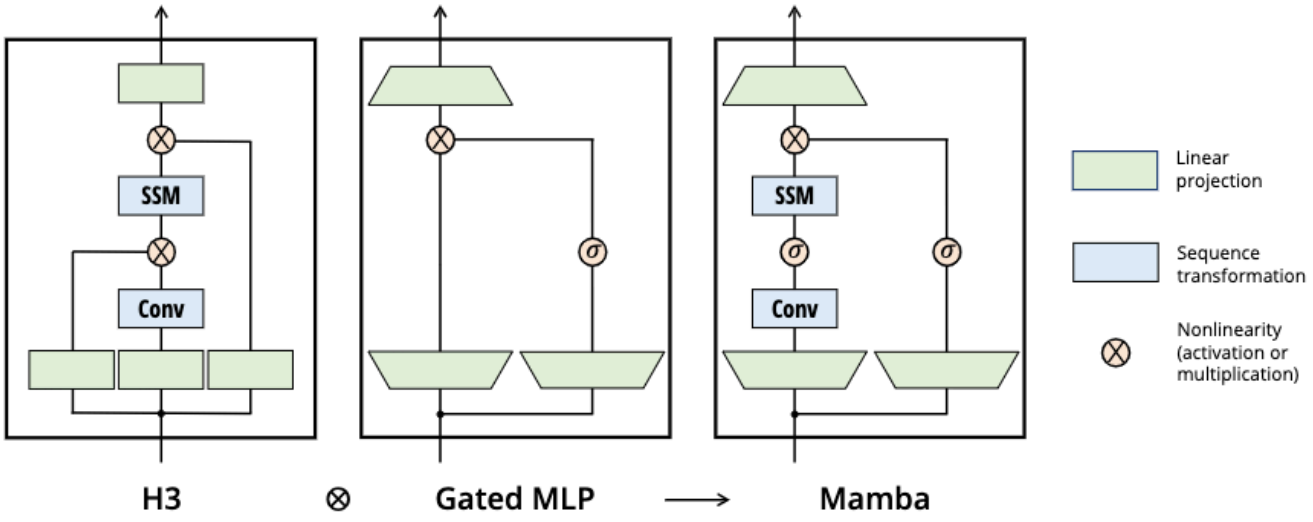
> △: discrete step size: 原文中说其 "controls the balance between how much to focus or ignore the current input x_t", 可以等价于RNN的gates, 越大的△越注意当下

The choice of s_Δ and τ_Δ is due to a connection to RNN gating mechanism

Efficient Implementation of Selective SSMs

主要谈利用GPU arch来完成selective scan: kernel fusion, parallel scan, and recomputation

A Simplified SSM Architecture



- Input: 2 x $D \to E \cdot D$, output: $E \cdot D \to D$, E=2
- Use SiLU / Swish activation function
- use an optional normalization layer

Properties of Selection Mechanisms

Δ: controls the balance between how much to focus or ignore the current input x_t. a large Δ resets the state h and focuses on the current input x, while a small Δ persists the state and ignores the current input.

Exps

Synthetic Tasks

Model	Arch.	Layer	Acc.
S4	No gate	S4	18.3
-	No gate	S6	**97.0**
H3	H3	S4	57.0
Hyena	H3	Hyena	30.1
-	H3	S6	**99.7**
-	Mamba	S4	56.4
-	Mamba	Hyena	28.4
Mamba	Mamba	S6	**99.8**

Table 1: (**Selective Copying.**) Accuracy for combinations of architectures and inner sequence layers.

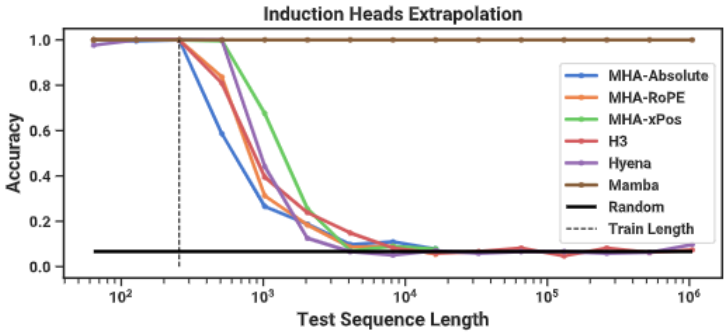


Table 2: (**Induction Heads.**) Models are trained on sequence length $2^8 = 256$, and tested on increasing sequence lengths of $2^6 = 64$ up to $2^{20} = 1048576$. Full numbers in Table 11.

Language Modeling

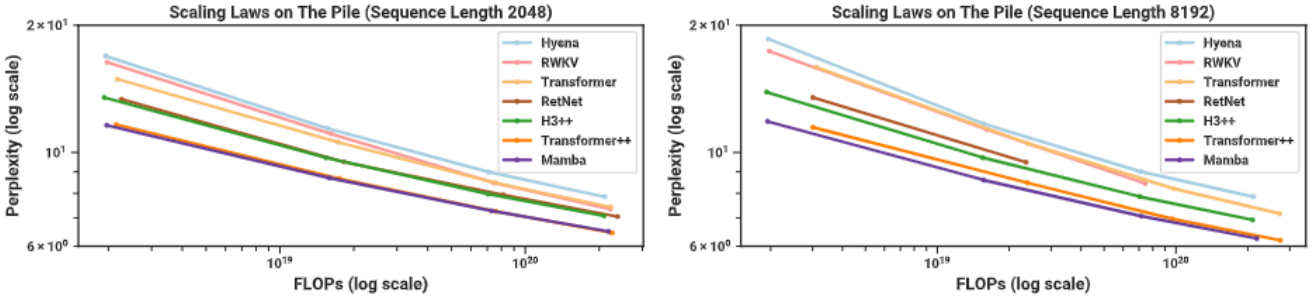


Figure 4: (**Scaling Laws.**) Models of size $\approx 125M$ to $\approx 1.3B$ parameters, trained on the Pile. Mamba scales better than all other attention-free models and is the first to match the performance of a very strong "Transformer++" recipe that has now become standard, particularly as the sequence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

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

Language Modeling Experiment Detail

```
* trained on 300B tokens on the Pile

| Parameters | Layers | Model dim. |
| ---------- | ------ | ---------- |
| 130M       | 24     | 768        |
| 370M       | 48     | 1024       |
| 790M       | 48     | 1536       |
| 1.4B       | 48     | 2048       |
| 2.8B       | 64     | 2560       |
```

DNA Modeling

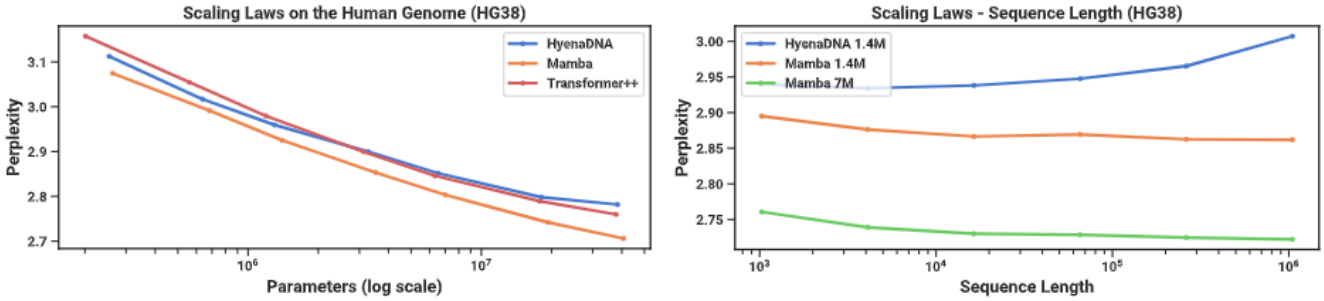


Figure 5: (**DNA Scaling Laws.**) Pretraining on the HG38 (human genome) dataset. (*Left*) Fixing short context length $2^{10} = 1024$ and increasing size from $\approx 200K$ to $\approx 40M$ parameters, Mamba scales better than baselines. (*Right*) Fixing model size and increasing sequence lengths while keeping tokens/batch and total training tokens fixed. Unlike baselines, the selection mechanism of Mamba facilitates better performance with increasing context length.

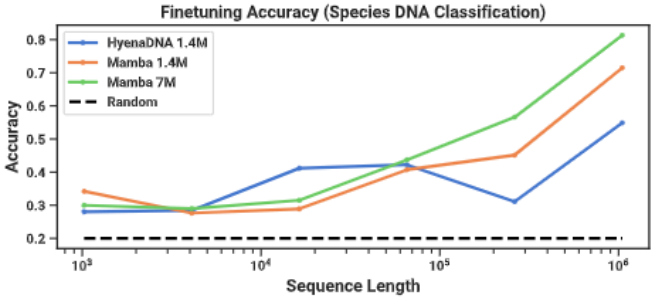


Figure 6: (**Great Apes DNA Classification.**) Accuracy after fine-tuning on sequences of length $2^{10} = 1024$ up to $2^{20} = 1048576$ using pretrained models of the same context length. Numerical results in Table 13.

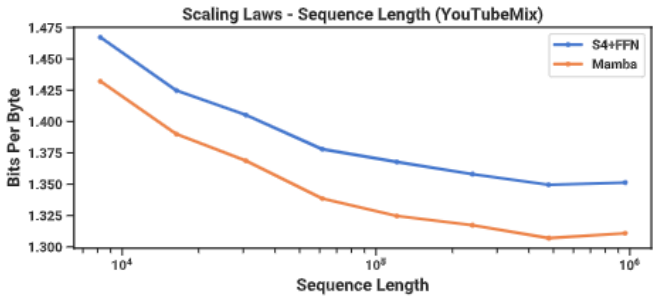


Figure 7: (**Audio Pretraining.**) Mamba improves performance over prior state-of-the-art (Sashimi) in autoregressive audio modeling, while improving up to minute-long context or million-length sequences (controlling for computation).

Audio

Autio Pretraining UP!

Autoregressive Speech Generation ⏬

Table 4: (**SC09**) Automated metrics for unconditional generation on a challenging dataset of fixed-length speech clips. (*Top to Bottom*) Autoregressive baselines, non-autoregressive baselines, Mamba, and dataset metrics.

Model	Params	NLL ↓	FID ↓	IS ↑	mIS ↑	AM ↓
SampleRNN	35.0M	2.042	8.96	1.71	3.02	1.76
WaveNet	4.2M	1.925	5.08	2.27	5.80	1.47
SaShiMi	5.8M	1.873	1.99	5.13	42.57	0.74
WaveGAN	19.1M	-	2.03	4.90	36.10	0.80
DiffWave	24.1M	-	1.92	5.26	51.21	0.68
+ SaShiMi	23.0M	-	1.42	5.94	69.17	0.59
Mamba	6.1M	**1.852**	0.94	6.26	88.54	0.52
Mamba	24.3M	1.860	**0.67**	**7.33**	**144.9**	**0.36**
Train	-	-	0.00	8.56	292.5	0.16
Test	-	-	0.02	8.33	257.6	0.19

Table 5: (**SC09 Model Ablations**) Models with 6M parameters. In SaShiMi's U-Net backbone, there are 8 center blocks operating on sequence length 1000, sandwiched on each side by 8 outer blocks on sequence length 4000, sandwiched by 8 outer blocks on sequence length 16000 (40 blocks total). The architecture of the 8 center blocks are ablated independently of the rest. Note that Transformers (MHA+MLP) were not tested in the more important outer blocks because of efficiency constraints.

Outer	Center	NLL ↓	FID ↓	IS ↑	mIS ↑	AM ↓
S4+MLP	MHA+MLP	1.859	1.45	5.06	47.03	0.70
S4+MLP	S4+MLP	1.867	1.43	5.42	53.54	0.65
S4+MLP	Mamba	1.859	1.42	5.71	56.51	0.64
Mamba	MHA+MLP	**1.850**	1.37	5.63	58.23	0.62
Mamba	S4+MLP	1.853	1.07	6.05	73.34	0.55
Mamba	Mamba	1.852	**0.94**	**6.26**	**88.54**	**0.52**

Speed and Memory Benchmarks

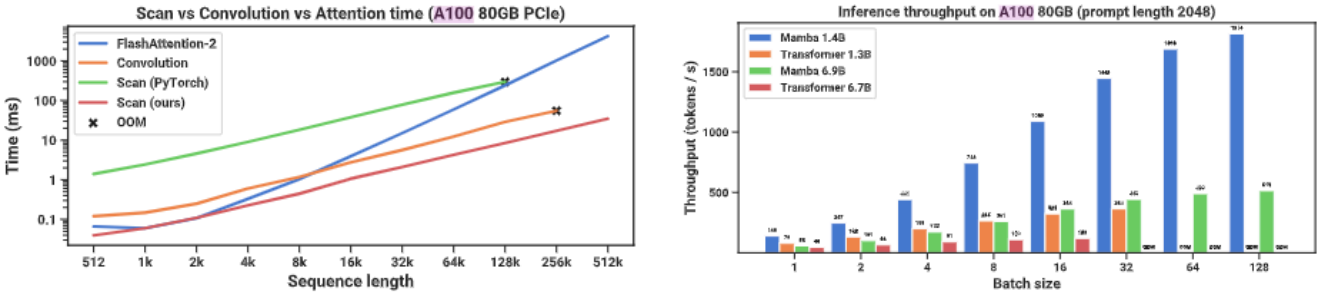
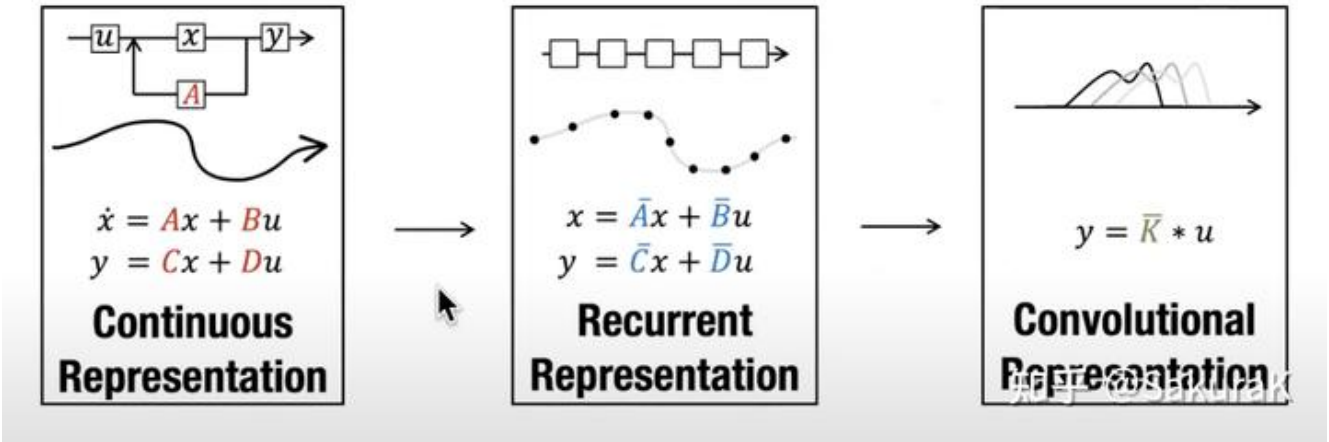


Figure 8: (**Efficiency Benchmarks.**) (*Left*) Training: our efficient scan is 40× faster than a standard implementation. (*Right*) Inference: as a recurrent model, Mamba can achieve 5× higher throughput than Transformers.

Links

Mamba Introduction

- 🌟🌟 visual guide to mamba and state space model [目前最清楚的mamba的visualization解释]
- 🌟Walk through the essence of Mamba architecture. 更偏向解释mamba的CUDA加速实现
- 🌟 The Annotated S4 https://srush.github.io/annotated-s4/ [强烈推荐 srush]
- 🚧 mamba-minimal: GitHub , pytorch实现[类比原repo中的CUDA实现，逻辑完全一致, 工程必看]
- 状态空间模型SSM的离散化过程推导 https://wuwenbinsights.com/posts/c8aa64ed/

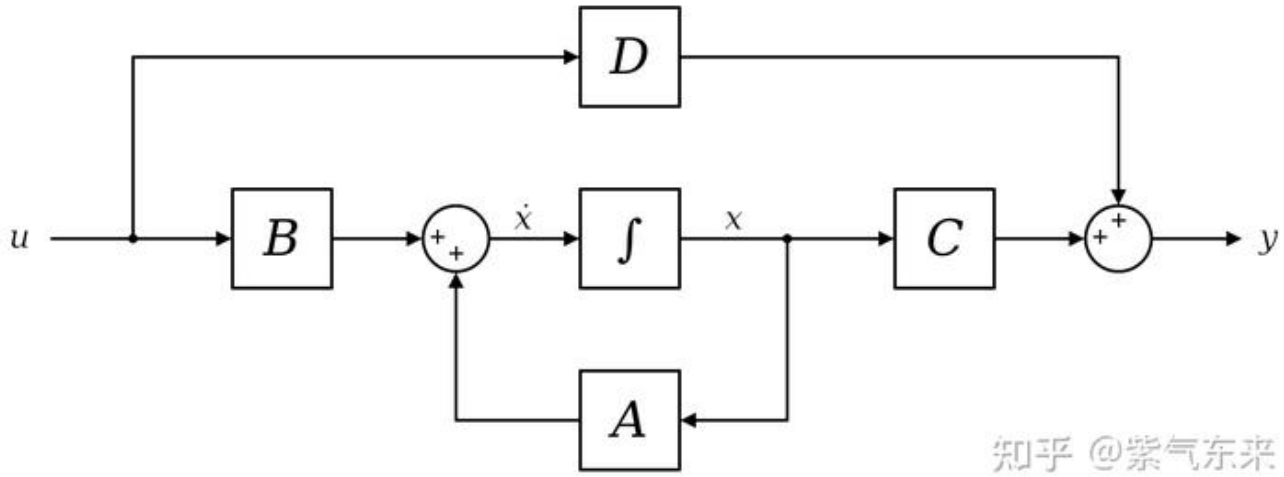


三种形式的特点：

- Continuous:
  - Pos: 自动处理连续数据(音频，时间序列)；可以进行数学上的分析精确计算
  - Neg: 训练和推理很缓慢
- Recurrent:
  - Pos: 对于序列数据的自然归纳，原则上无上下文限制；高效推理(恒定时间状态更新)
  - Neg: 学习缓慢（缺乏并行性）训练过长序列时梯度消失或爆炸
- Convolution
  - Pos: 本地的、可解释的特征(A/B/C/D learnable), 高效（可并行化）训练

    - Neg: 固定上下文大小, 在线或自回归上下文中的速度较慢（必须重新计算每个新数据点的整个输入）

- 另一种推理的参考 https://zhuanlan.zhihu.com/p/680833040 从控制论出发



连续状态的时不变系统LTI

Other links

- Awesome mamba papers (多杂而无用，纯属罗列)