Linear Transformer And Linear Attention

Nasy

Apr 12, 2024

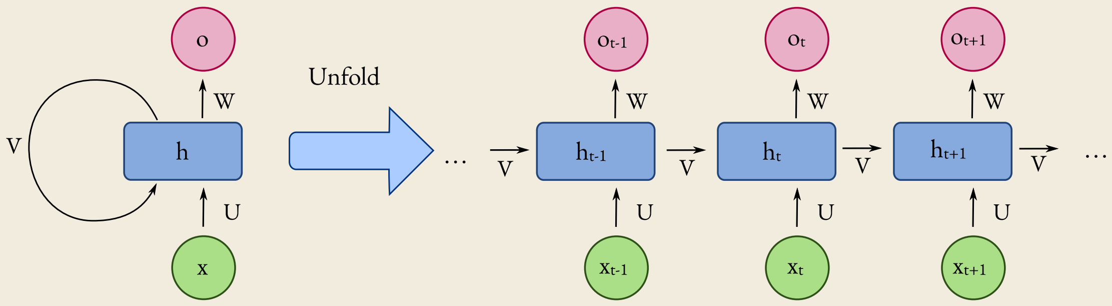
Table of Contents

Introduction

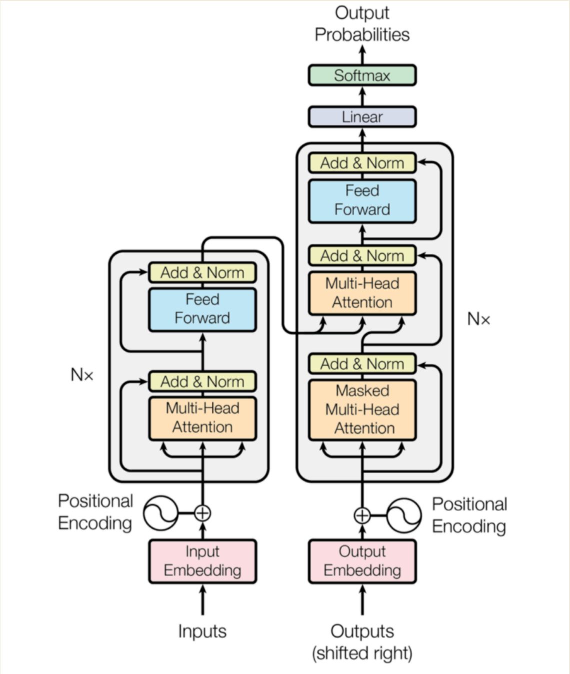
RNN And Transformer Comparison

- RNN:
  - Train: **slow**
  - Inference: **fast**
- Transformer:
  - Train: **fast**
  - Inference: **slow**

RNN And Transformer

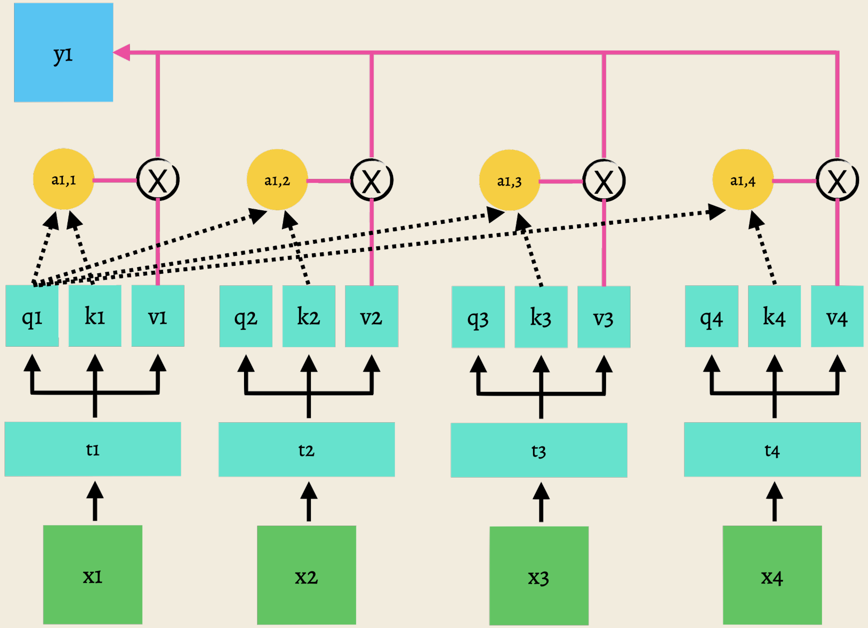


RNN: A fully recurrent network



Transformer model archiecture.

Why Is Transformer Inference Slow?



$$
\begin{aligned}
a_{1,i} &= \mathrm{softmax}\left(\frac{q_1 \cdot k_i}{\sqrt{d}}\right) \\
y_1 &= \textstyle\sum_i a_{1,i} v_i \\
QK &\rightarrow O(n^2 d) \\
\mathrm{softmax} &\rightarrow O(n^2) \\
\textstyle\sum &\rightarrow O(n^2 d)
\end{aligned}
$$

Linear Transformer

Papers in Linear Transformer

Linear Transformer Katharopoulos, Angelos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret, "Transformers Are Rnns: Fast Autoregressive Transformers with Linear Attention", 2020

AFT Zhai, Shuangfei, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh, Ruixiang Zhang, and others, "An Attention Free Transformer", 2021

RWKV Peng, Bo, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, and others, "RWKV: Reinventing Rnns for the Transformer Era", 2023

Transformers

$$Y = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \quad (1)$$

$$Y_i = \frac{\sum_{j=1}^{N} \exp(q_i \cdot k_j) V_j}{\sum_{j=1}^{N} \exp(q_i \cdot k_j)} \quad (2)$$

$$Y_i = \frac{\sum_{j=1}^{N} \text{sim}(q_i, k_j) V_j}{\sum_{j=1}^{N} \text{sim}(q_i, k_j)} \quad (3)$$

The "sim" function is a similarity function.

If $\text{sim}(q, k) = \text{softmax}\left(\frac{q^T \cdot k}{\sqrt{d}}\right)$, Equation 2 is equivalent to Equation 3.

Linearized Transformer

If we have a kernel function φ, and let

$$\mathrm{sim}(q_i, k_i) = \varphi(q_i)\varphi(k_i) \tag{4}$$

then,

$$Y_i = \frac{\sum_{j=1}^{N} \varphi(q_i)\varphi(k_j)^T V_j}{\sum_{j=1}^{N} \varphi(q_i)\varphi(k_j)^T} = \frac{\varphi(q_i)\sum_{j=1}^{N} \varphi(k_j)^T V_j}{\varphi(q_i)\sum_{j=1}^{N} \varphi(k_j)^T} \tag{5}$$

We can write it in vectorized,

$$\underbrace{\left(\varphi(Q)\varphi(K)^T\right)V}_{\text{Traditional Transformer } (O(n^2 d))} = \overbrace{\varphi(Q)\left(\varphi(K)^T V\right)}^{\text{Linearized Transformer } (O(nd^2))} \tag{6}$$

Inference

We can expend the sum part in Equation 5, and it is a recurrent formula:

$$
\begin{aligned}
S_i &= \sum_{j=1}^{i} \varphi(k_j)^T V_j = \varphi(k_i)^T + \sum_{j=1}^{i-1} \varphi(k_j)^T V_j \ = \varphi(k_i)^T + S_{i-1} \\
Z_i &= \sum_{j=1}^{i} \varphi(k_j)^T = \varphi(k_i)^T + \sum_{j=1}^{i-1} \varphi(k_j)^T \qquad = \varphi(k_j)^T + Z_{i-1}
\end{aligned}
\tag{7}
$$

Here, we can regard S_i and Z_i as a state, thus, we can reuse them.

Attention Free Transformer

$$Y_i = \sigma_q(Q_i) \odot \frac{\sum_{j=1}^{N} \exp\left(K_j + w_{i,j}\right) \odot V_j}{\sum_{j=1}^{N} \exp\left(K_j + w_{i,j}\right)} \tag{8}$$

where, σ is sigmoid function.

It is a multi-head attention w/ heads equal to the dimension of embedding. The time complexity is

Attention Free Transformer

$$Y_i = \sigma_q(Q_i) \odot \frac{\sum_{j=1}^{N} \exp\left(K_j + w_{i,j}\right) \odot V_j}{\sum_{j=1}^{N} \exp\left(K_j + w_{i,j}\right)} \tag{9}$$

where, σ is sigmoid function.

It is a multi-head attention w/ heads equal to the dimension of embedding. The time complexity is $O(n^2 d)$.

Linear Time Attention Free Transformer

AFT-local

$$w_{i,j} = \begin{cases} w_{i,j} & \text{if } |i - j| < s \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $s < n$ is a local window size. The time complexity is $O(nsd)$, $s < n$.

Linear Time Attention Free Transformer

AFT-local

$$w_{i,j} = \begin{cases} w_{i,j} & \text{if } |i-j| < s \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where $s < n$ is a local window size. The time complexity is $O(nsd)$, $s < n$.

AFT-simple

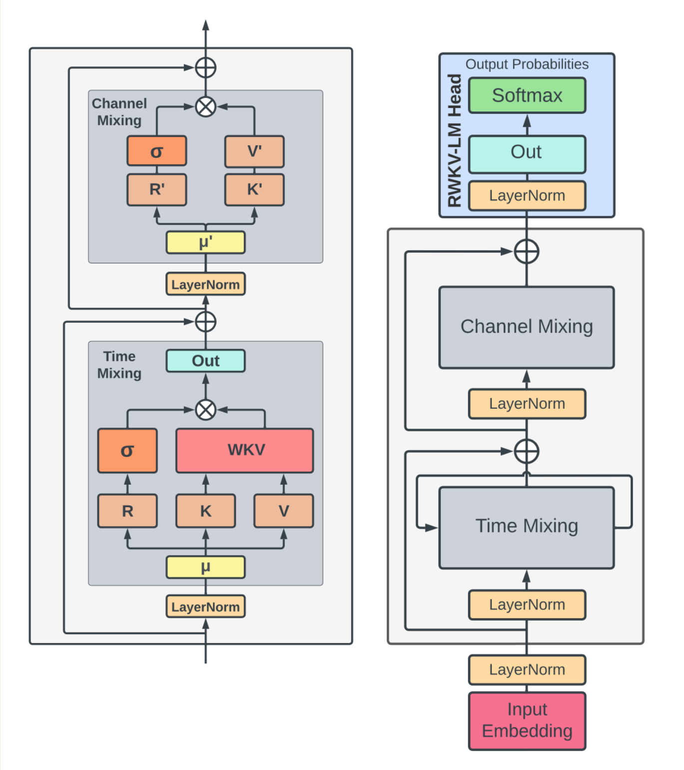
Drop the $w_{i,j}$ term in AFT. The time complexity is $O(nd)$.

$$Y_i = \sigma_q(Q_i) \odot \frac{\sum_{j=1}^{N} \exp(K_j) \odot V_j}{\sum_{j=1}^{N} \exp(K_j)} = \sigma_q(Q_i) \odot \sum_{j=1}^{N} (\text{softmax}(K) \odot V)_j \tag{13}$$

RWKV

- R: The **Receptance** vector acts as the receiver of past information.
- W: The **Weight** signifies the positional weight decay vector, a trainable parameter within the model.
- K: The **Key** vector performs a role analogous to K in traditional attention mechanisms.
- V: The **Value** vector functions similarly to V in conventional attention processes.

RWKV



$$
\begin{aligned}
r_t &= W_r \cdot (\mu_r \odot x_t + (1 - \mu_r) \odot x_{t-1}) \\
k_t &= W_k \cdot (\mu_k \odot x_t + (1 - \mu_k) \odot x_{t-1}) \\
v_t &= W_v \cdot (\mu_v \odot x_t + (1 - \mu_v) \odot x_{t-1})
\end{aligned} \tag{14}
$$

$$
wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} \odot v_i + e^{u+k_t} \odot v_t}{e^{-(t-1-i)w+k_i} + e^{u+k_t}} \tag{15}
$$

$$
o_t = W_o \cdot (\sigma(r_t) \odot wkv_t) \tag{16}
$$

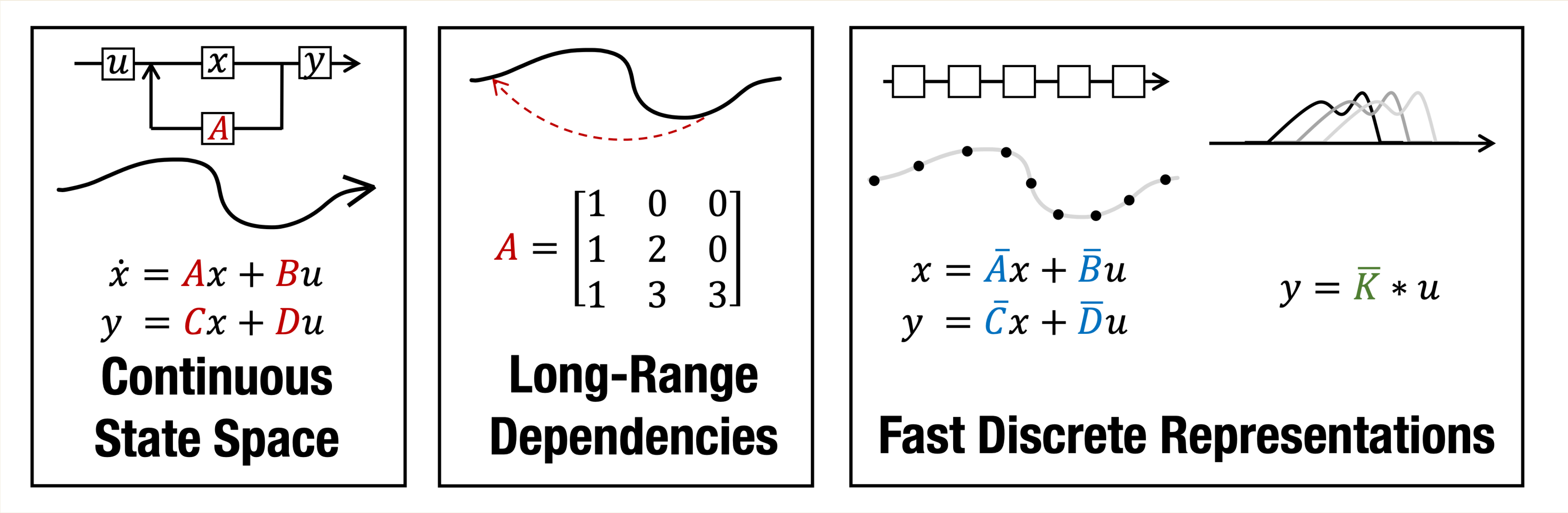
Mamba

Papers

SSM Gu, Albert, Karan Goel, and Christopher Ré, "Efficiently Modeling Long Sequences with Structured State Spaces", 2021

mamba Gu, Albert, and Tri Dao, "Mamba: Linear-Time Sequence Modeling with Selective State Spaces", 2023

State Spaces Model (SSM) Framework



State Spaces Model (SSM)

$$
\begin{aligned}
x'(t) &= \boldsymbol{A}x(t) + \boldsymbol{B}u(t) \\
y(t) &= \boldsymbol{C}x(t) + \boldsymbol{D}u(t)
\end{aligned}
\tag{17}
$$

$\boldsymbol{u(t)}$ the $1-D$ input.

$\boldsymbol{x(t)}$ the $N-D$ latent state.

$\boldsymbol{y(t)}$ the $1-D$ output.

$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ the model parameters.

Discretization

$$\begin{aligned} x'(t) &= \bar{\boldsymbol{A}} x_{t-1} + \bar{\boldsymbol{B}} u_t \\ y(t) &= \bar{\boldsymbol{C}} x_t + \bar{\boldsymbol{D}} u_t \end{aligned} \tag{18}$$

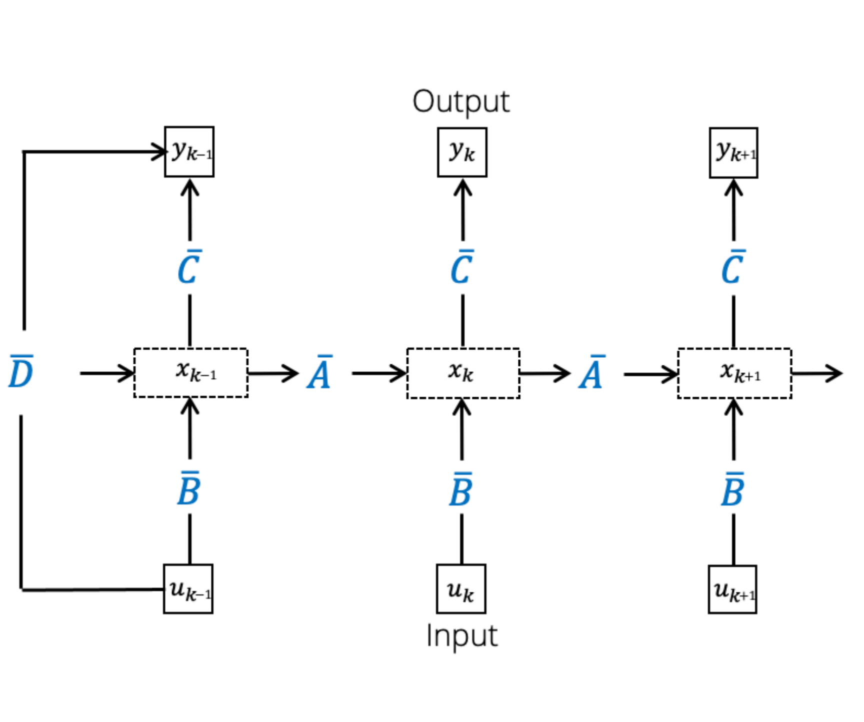
where in the *S4* model,

$$\begin{aligned} \bar{\boldsymbol{A}} &= \left(\boldsymbol{I} - \frac{\Delta}{2} \cdot \boldsymbol{A}\right)^{-1} \left(\boldsymbol{I} + \frac{\Delta}{2} \cdot \boldsymbol{A}\right) \\ \bar{\boldsymbol{B}} &= \left(\boldsymbol{I} - \frac{\Delta}{2} \cdot \boldsymbol{B}\right)^{-1} \Delta \boldsymbol{B} \\ \bar{\boldsymbol{C}} &= C \\ \bar{\boldsymbol{D}} &= D \end{aligned} \tag{19}$$

Discretization in Mamba

$$
\begin{aligned}
\bar{A} &= \exp(\Delta A) \\
\bar{B} &= (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B
\end{aligned}
\tag{20}
$$

Recurrent and Convolution



$$x_0 = \bar{B}u_0;\ x_1 = \bar{A}\bar{B}u_0 + \bar{B}u_1;$$

$$x_2 = \bar{B}u_2 = \bar{A}^2\bar{B}u_0 + \bar{A}\bar{B}u_1 + \bar{B}u_2$$

$$\dots$$

$$y_0 = \bar{C}\bar{B}u_0 + \bar{D}u_0$$

$$y_1 = \bar{C}\bar{A}\bar{B}u_0 + \bar{C}\bar{B}u_1 + \bar{D}u_1$$

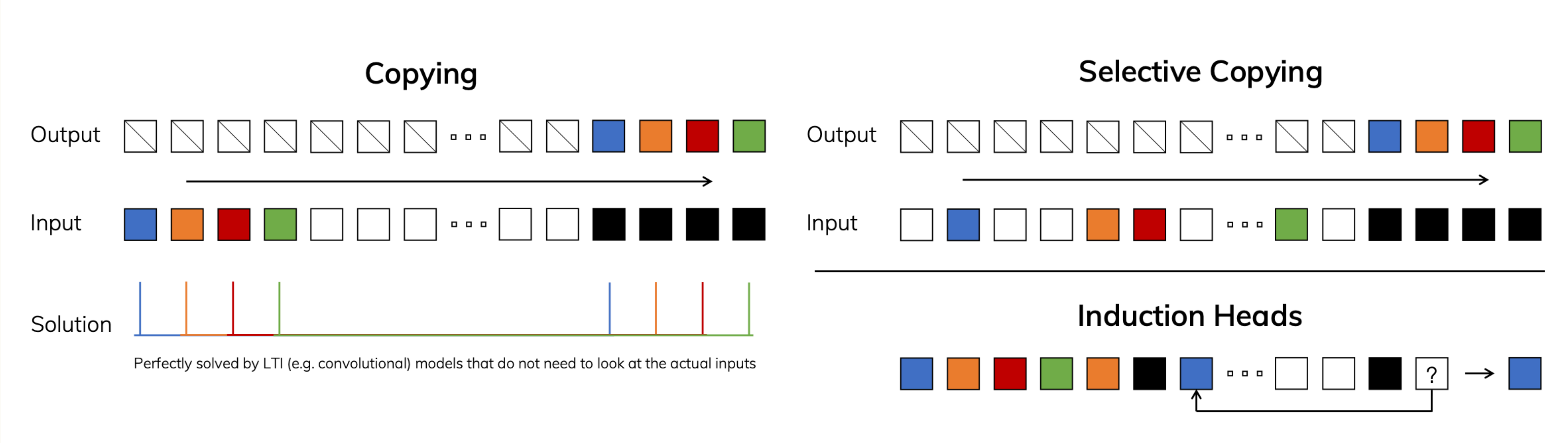
$$y_2 = \bar{C}\bar{A}^2\bar{B}u_0 + \bar{C}\bar{A}\bar{B}u_1 + \bar{C}\bar{B}u_2 + \bar{D}u_2$$

$$\dots$$

$$\bar{K} = \left(\bar{C}\bar{A}^i\bar{B}\right)_{i\in[L]} = \left(\bar{C}\bar{B}, \bar{C}\bar{A}\bar{B}, \dots, \bar{C}\bar{A}^{L-1}\bar{B}\right)$$

$$y = \bar{K} * u$$

Selection


Copying
Output
Input
Solution
Perfectly solved by LTI (e.g. convolutional) models that do not need to look at the actual inputs
Selective Copying
Output
Input
Induction Heads
?

SSM And SSM w/ Selection

Algorithm 1 SSM (S4)

Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$

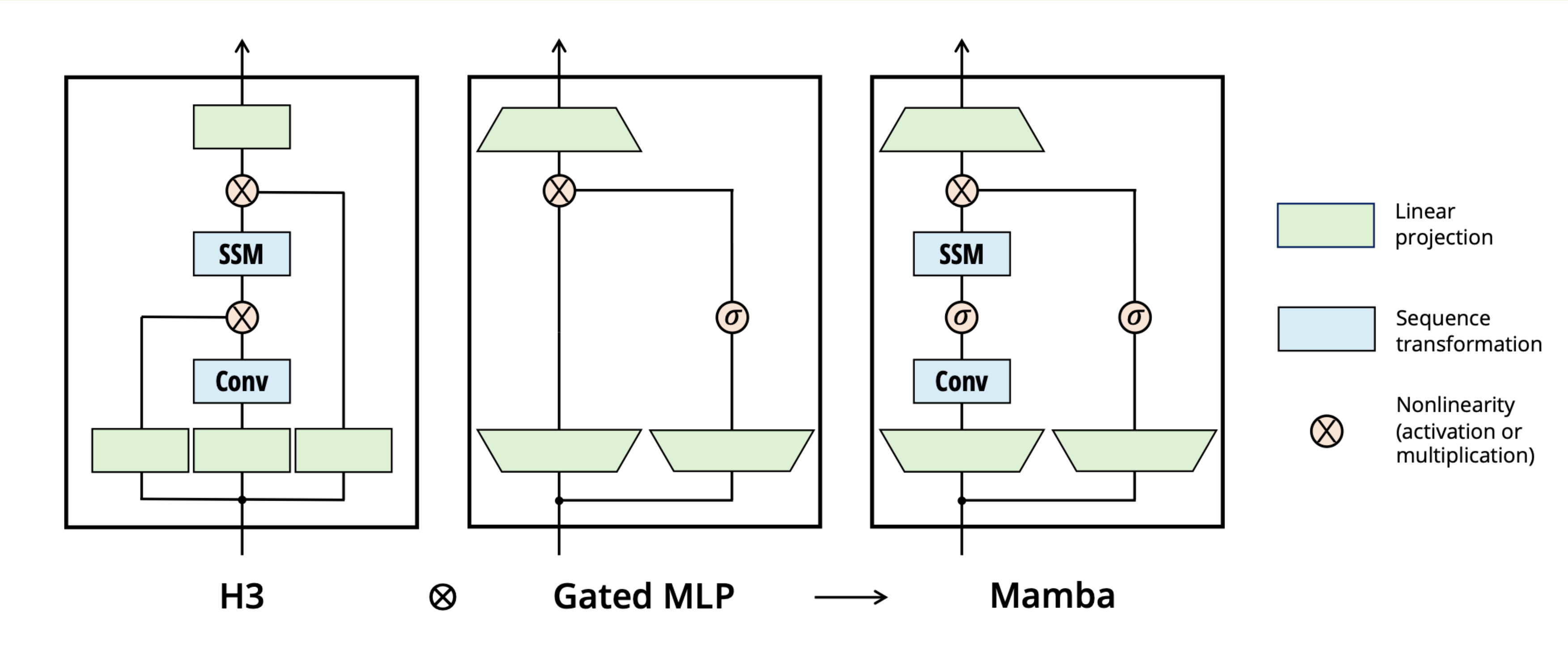
$\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
▷ Represents structured $N \times N$ matrix
$\boldsymbol{B} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\boldsymbol{C} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\Delta : (\mathtt{D}) \leftarrow \tau_\Delta(\mathsf{Parameter})$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

Algorithm 2 SSM + Selection (S6)

Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$

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

Mamba


SSM
Conv
σ
Linear projection
Sequence transformation
Nonlinearity (activation or multiplication)
H3
⊗
Gated MLP
Mamba

Mamba Results – Selection

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.

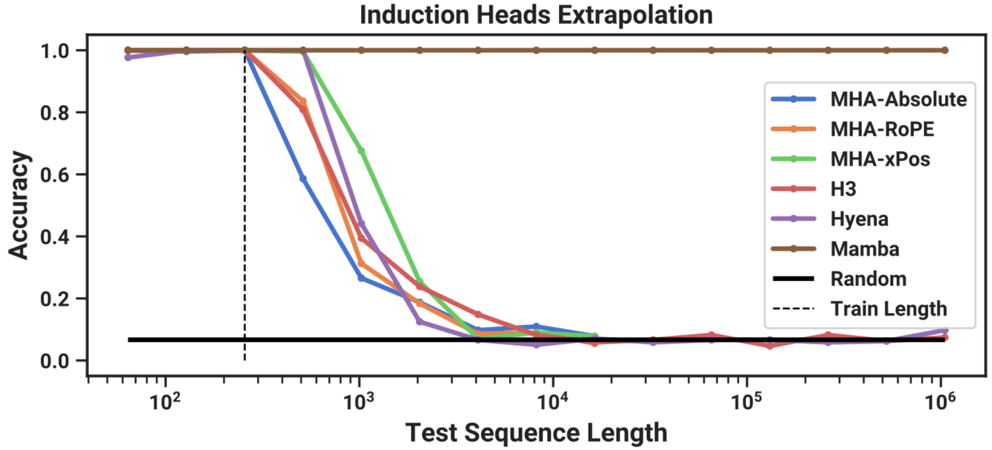


Table 2: (**Induction Heads**.) Models are trained on sequence length $2^8 = 256$, and tested on increasing sequence lengths of $2^6 = 64$ up to $2^{20} = 1048576$. Full numbers in Table 11.

Mamba Results – Scaling Laws

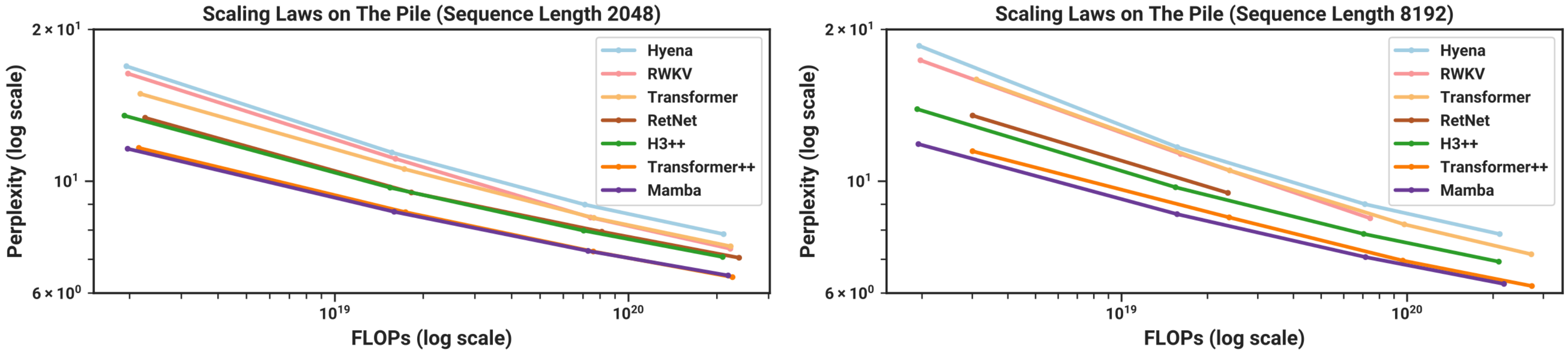


Figure 4: (**Scaling Laws.**) Models of size $\approx 125M$ to $\approx 1.3B$ parameters, trained on the Pile. Mamba scales better than all other attention-free models and is the first to match the performance of a very strong "Transformer++" recipe that has now become standard, particularly as the sequence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

Mamba Results – Zero shot

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

Mamba Results – Efficency Benchmarks

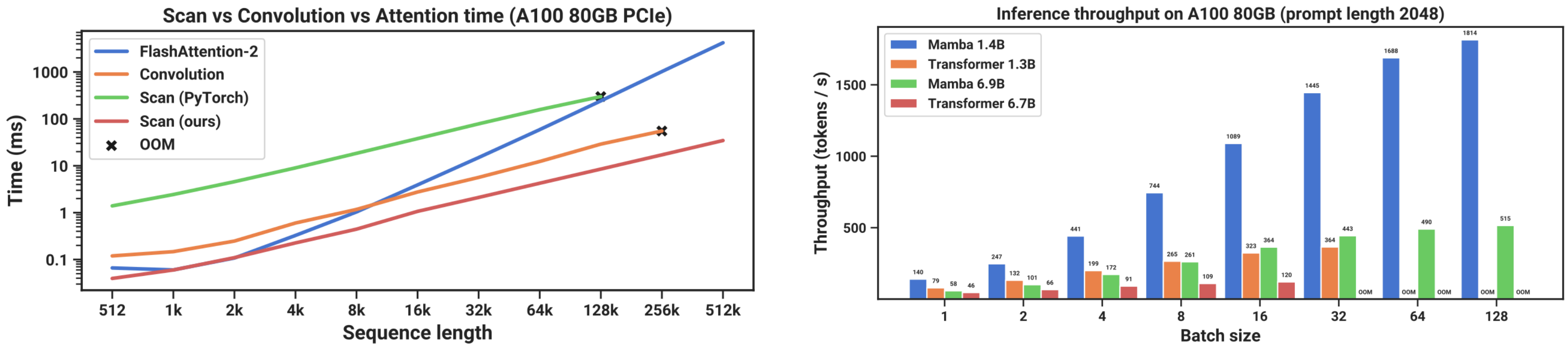


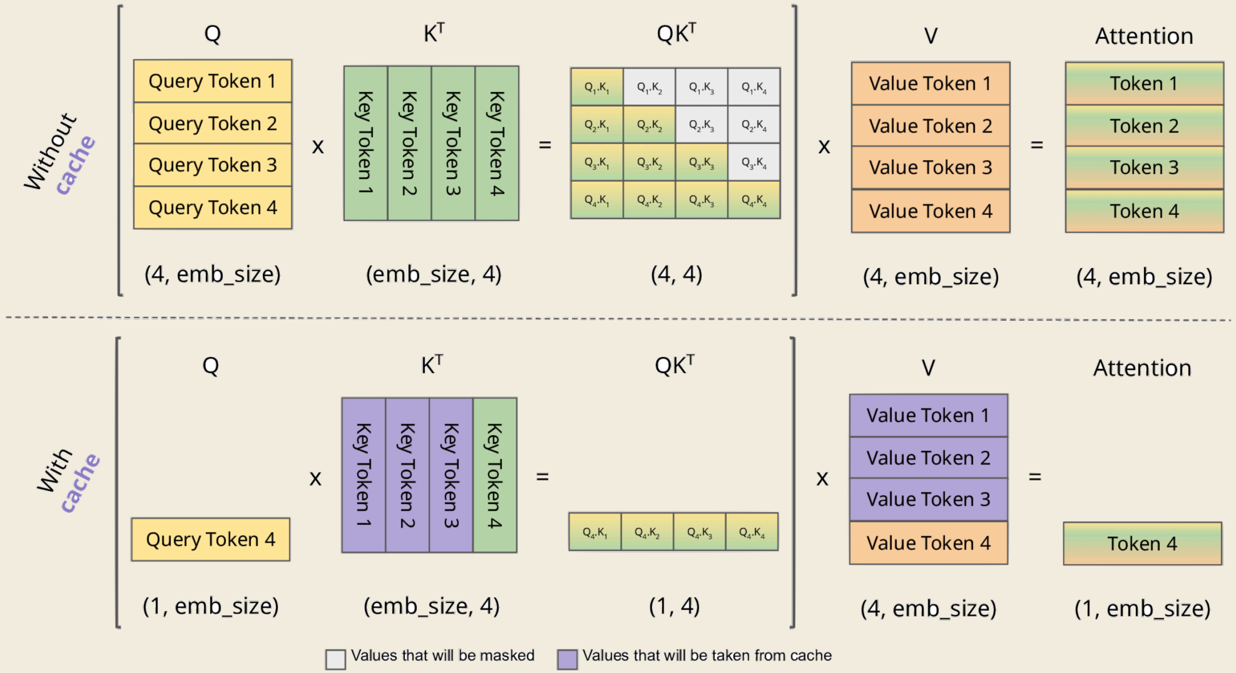
Figure 8: (**Efficiency Benchmarks.**) (*Left*) Training: our efficient scan is 40× faster than a standard implementation. (*Right*) Inference: as a recurrent model, Mamba can achieve 5× higher throughput than Transformers.

More Results And Ablation Studies

For the complete results and ablation studies,
please see the Paper: https://arxiv.org/abs/2312.00752

Linear Attention

KV Cache



Pope, Reiner, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, and others, “Efficiently Scaling Transformer Inference”, 2022

Conclusion

Summary

- RNN And Transformer
- Linear Transformer
  - Linearized Transformers $Q \cdot (KV)$
  - Attention Free Transformer
  - RWKV
  - SSM And Mamba
- KV Cache

Bibliography

Gu, Albert, and Tri Dao, "Mamba: Linear-Time Sequence Modeling with Selective State Spaces", 2023

Gu, Albert, Karan Goel, and Christopher Ré, "Efficiently Modeling Long Sequences with Structured State Spaces", 2021

Katharopoulos, Angelos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret, "Transformers Are Rnns: Fast Autoregressive Transformers with Linear Attention", 2020

Peng, Bo, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, and others, "RWKV: Reinventing Rnns for the Transformer Era", 2023

Pope, Reiner, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, and others, "Efficiently Scaling Transformer Inference", 2022

Zhai, Shuangfei, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh, Ruixiang Zhang, and others, "An Attention Free Transformer", 2021