IERG5050 AI Foundation Models, Systems and Applications
Spring 2025

Beyond Transformers: Alternative Architectures for LLMs
SSMs, MAMBA, RWKV

Prof. Wing C. Lau
wclau@ie.cuhk.edu.hk
http://www.ie.cuhk.edu.hk/wclau

Acknowledgements

Many of the slides in this lecture are adapted from the sources below. Copyrights belong to the original authors.

- UC Berkeley CS294-162: AI-Systems (LLM Edition), Fall 2023, by Profs. Joseph E. Gonzalez and Matei Zaharia, https://learning-systems.notion.site/AI-Systems-LLM-Edition-294-162-Fall-2023-661887583bd340fa851e6a8da8e29abb
- Shubham Gupta, "State Space Models 101", https://github.com/goodhamgupta/ssm_101/blob/main/ssm_101.pptx
- Stanford CS336: Language Modeling from Scratch, Spring 2024
  - by Profs. Tatsunori Hashimoto, Percy Liang, https://stanford-cs336.github.io/spring2024/
- Stanford CS229S: Systems for Machine Learning, Fall 2023
  by Profs. Azalia Mirhoseini, Simran Arora, https://cs229s.stanford.edu/fall2023/
- CMU 11-667: Large Language Models: Methods and Applications, Fall 2024
  by Profs. Chenyan Xiong and Daphne Ippolito, https://cmu-llms.org
- CMU 11-711: Advanced Natural Language Processing (ANLP), Spring 2024
  by Prof. Graham Neubig, https://phontron.com/class/anlp2024/lectures/
- UPenn CIS7000: Large Language Models, Fall 2024
  by Prof. Mayur Naik, https://llm-class.github.io/schedule.html
- UWaterloo CS886: Recent Advances on Foundation Models, Winter 2024
  by Prof. Wenhu Chen, https://cs.uwaterloo.ca/~wenhuche/teaching/cs886/
- MIT 6.5940: TinyML and Efficient Deep Learning Computing, Fall 2024
  by Prof. Song Han, https://hanlab.mit.edu/courses/2024-fall-65940
- UMD CMSC848K: Multimodal Foundation Models, Fall 2024
  by Prof. Jia-Bin Huang, https://jbhuang0604.github.io/teaching/CMSC848K/

State Space Models (SSMs), Selective State Space Models (SSSMs) and MAMBA

High-level Summary Video:
MAMBA and State Space Models Explained | SSM Explained
by AI Coeffee Break with Letitia
https://www.youtube.com/watch?v=vrF3MtGwD0Y

Detailed Insights by the Inventor: Prof. Albert GU,
On the Tradeoffs of State Space Models
https://simons.berkeley.edu/talks/albert-gu-carnegie-mellon-university-2024-09-27

https://icml.cc/virtual/2024/39087

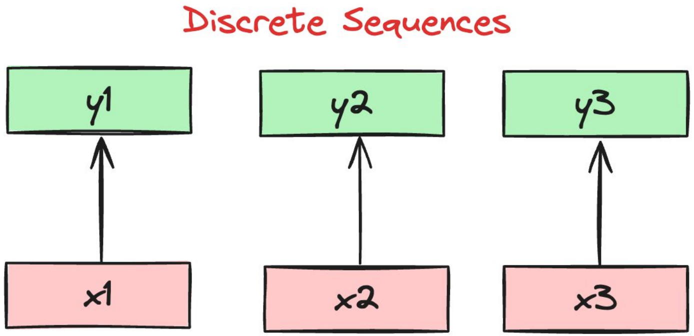
State Space Models 101

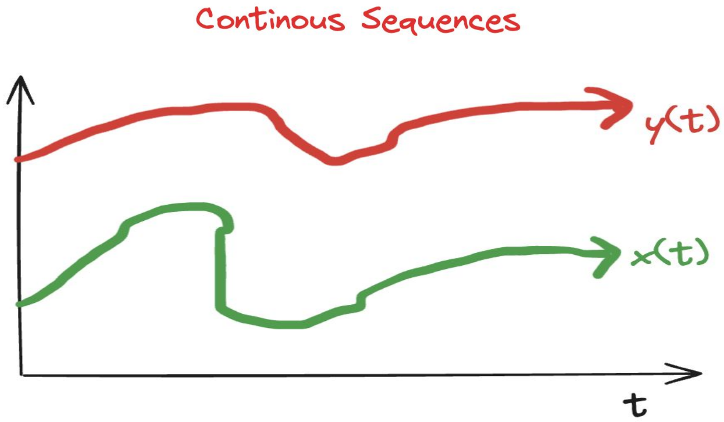
Shubham Gupta

in/shubhamgupta2208 shubhamg.in

Sequence Modelling: Recap

Goal: Map an input sequence to an output sequence





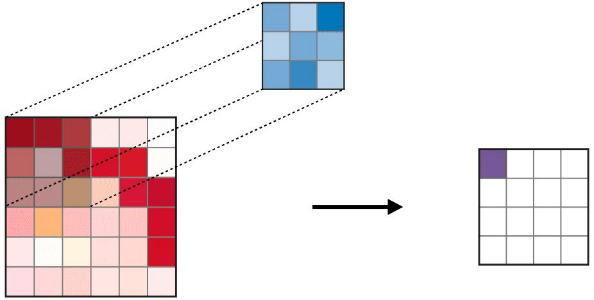
Sequence Modelling: Models

RNN

Source: Medium

Training: O(N) $\nparallel$
Inference: O(N)
Performance: ☹

CNN

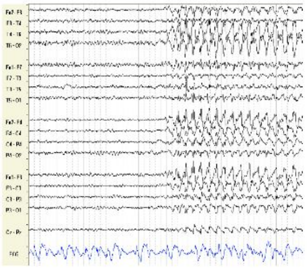


Source: Medium

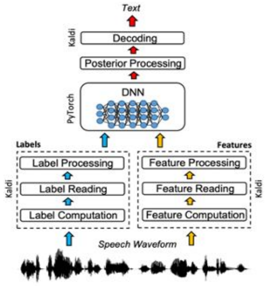
Training: O(N) $\parallel$
Inference: O(N)
Performance: ☹

Sequence Modelling: Transformers

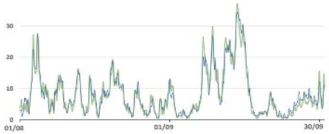
- ➢ SoTA on sequence modelling tasks
- ➢ Struggle to scale over long sequences
- ➢ Why do we care?
  - ○ Enable new capabilities
  - ○ Model other sequential data



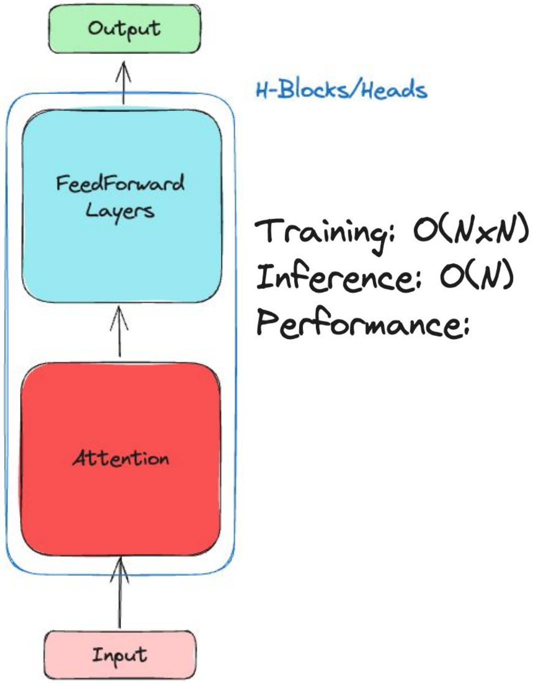
EEG/ECG



Audio



Energy Forecasting



Long Range Arena Benchmark

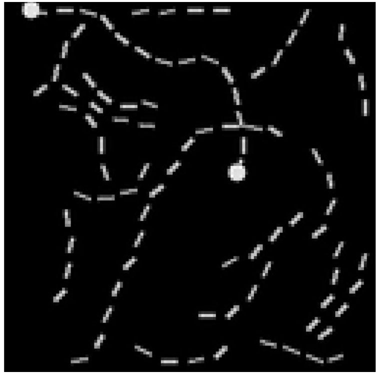
- ➢ LRA [Yi Tay et al., 2020]
- ➢ Measure long-context model quality
- ➢ Multiple input modalities
- ➢ Total tasks: 6
- ➢ Input sequence length: 1k-16k
- ➢ Transformer Performance:
  - ○ Avg. Accuracy: **52%** 👎
  - ○ Unable to solve Path-X

ListOps

[MAX 4 3 [MIN 2 3] 1 0 [MEDIAN 1 5 8 9, 2]]

Output: 5

Path-Finding



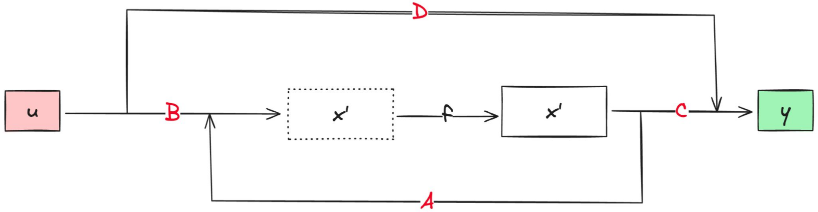
Ideal Model

Training: O(N)
Inference: O(N)
Performance:

Obtain best of all models for long-context?

State Space Models

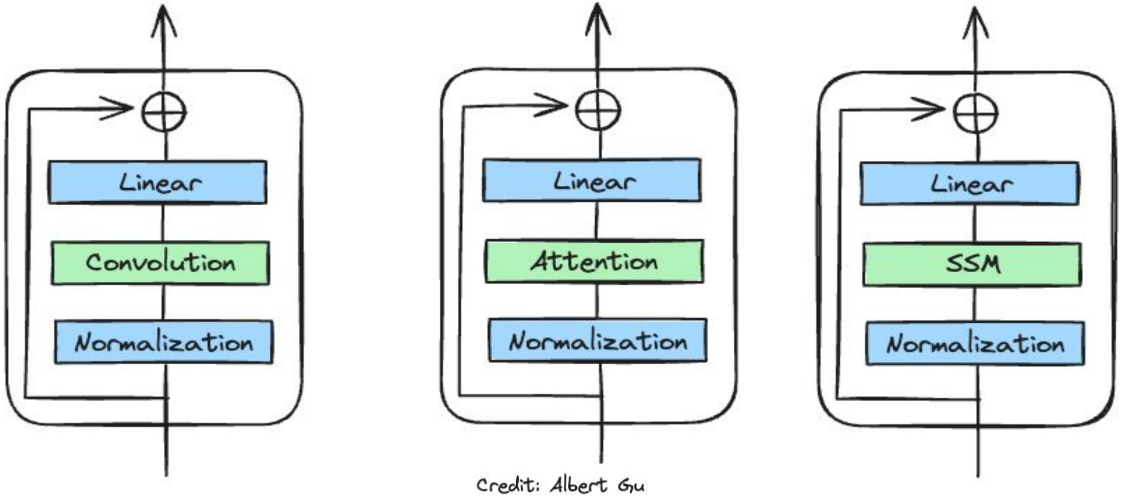
- ➢ Kalman, 1960
- ➢ Used in control theory/signal processing
- ➢ Modelled as *continuous-timed* differential equation
- ➢ 1-layer, Linear Model
- ➢ Time - Invariant



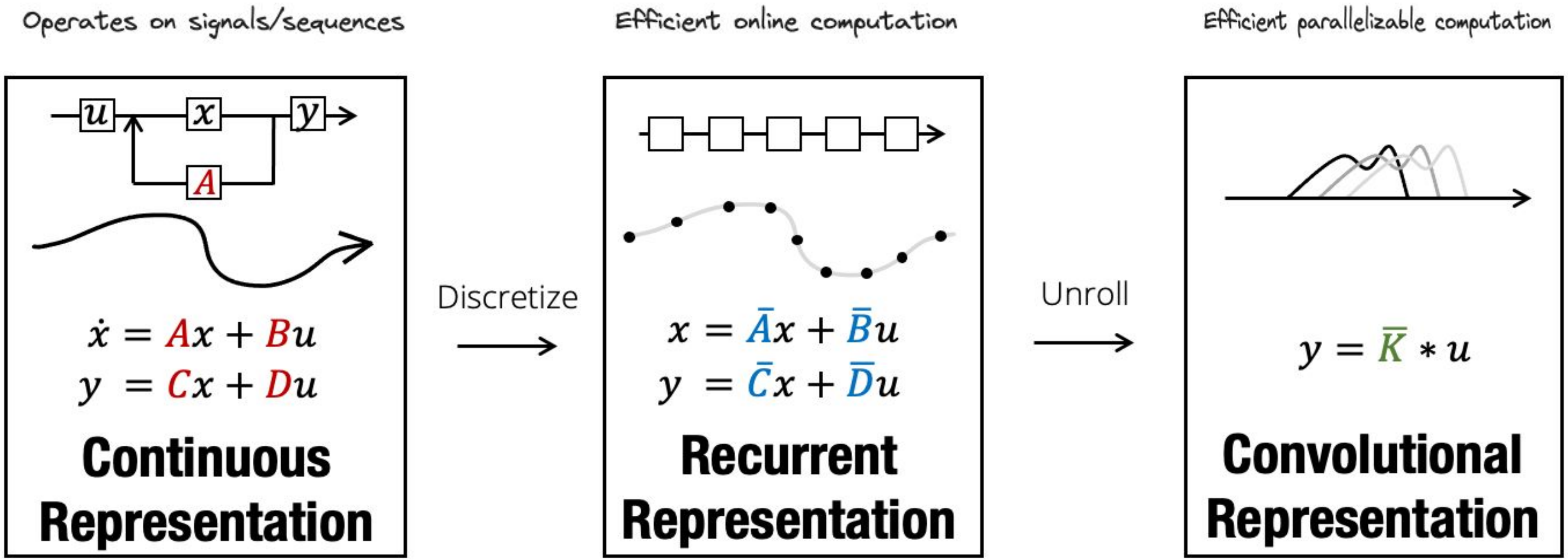
$$x'(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx'(t) + Du(t)$$

Deep SSMs

- ➢ Deep, non-linear model
- ➢ Deterministic transformation



SSM Properties



Source: Albert Gu

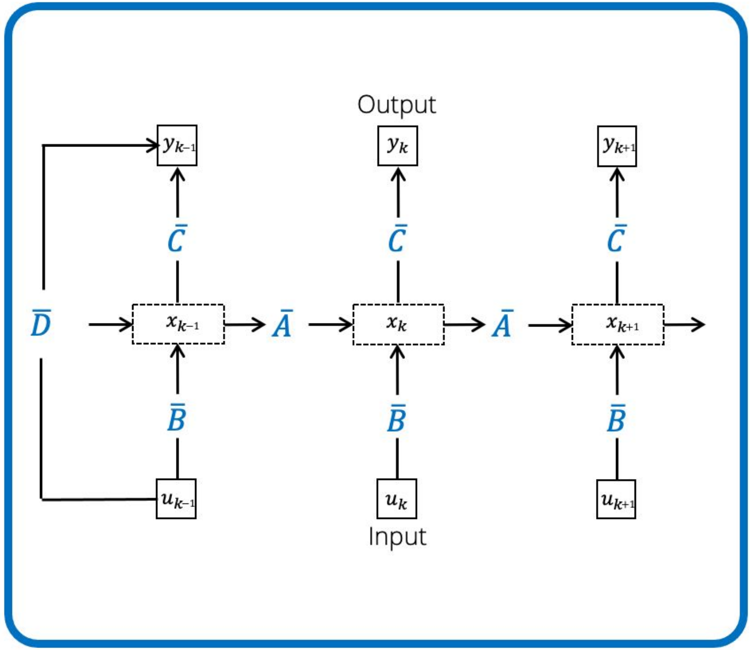
SSM: Recurrent

$$x'(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx'(t) + Du(t)$$

Discretize: $\bar{A} = I + \Delta A$

Recurrent "hidden state" $x_k = \bar{A}x_{k-1} + \bar{B}u_k$

Out Projection $y_k = \bar{C}x_k + \bar{D}u_k$



Source: Albert Gu

SSM: Convolutional

$$x_k = \bar{A}x_{k-1} + \bar{B}u_k \qquad y_k = \bar{C}x_k + \bar{D}u_k$$

Expand the terms

$$x_0 = \bar{B}u_0 \quad x_1 = \bar{A}\bar{B}u_0 + \bar{B}u_1 \quad x_2 = \bar{A}^2\bar{B}u_0 + \bar{A}\bar{B}u_1 + \bar{B}u_2$$

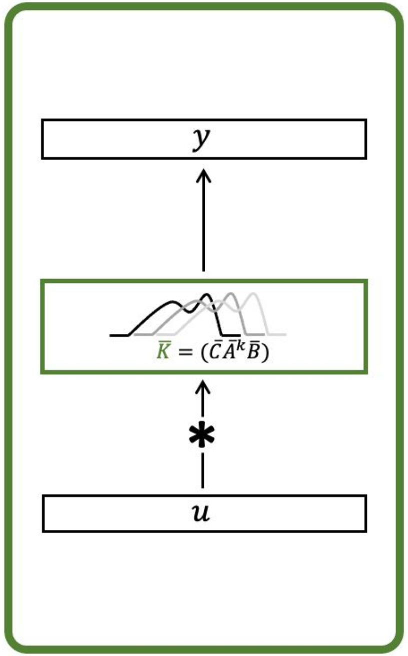
Output will be a linear projection of state

$$y_0 = \bar{C}\bar{B}u_0 + \bar{D}u_0$$
$$y_1 = \bar{C}\bar{A}\bar{B}u_0 + \bar{C}\bar{B}u_1 + \bar{D}u_1$$
$$y_2 = \bar{C}\bar{A}^2\bar{B}u_0 + \bar{C}\bar{A}\bar{B}u_1 + \bar{C}\bar{B}u_2 + \bar{D}u_2$$

No non-linearity => Simple computation

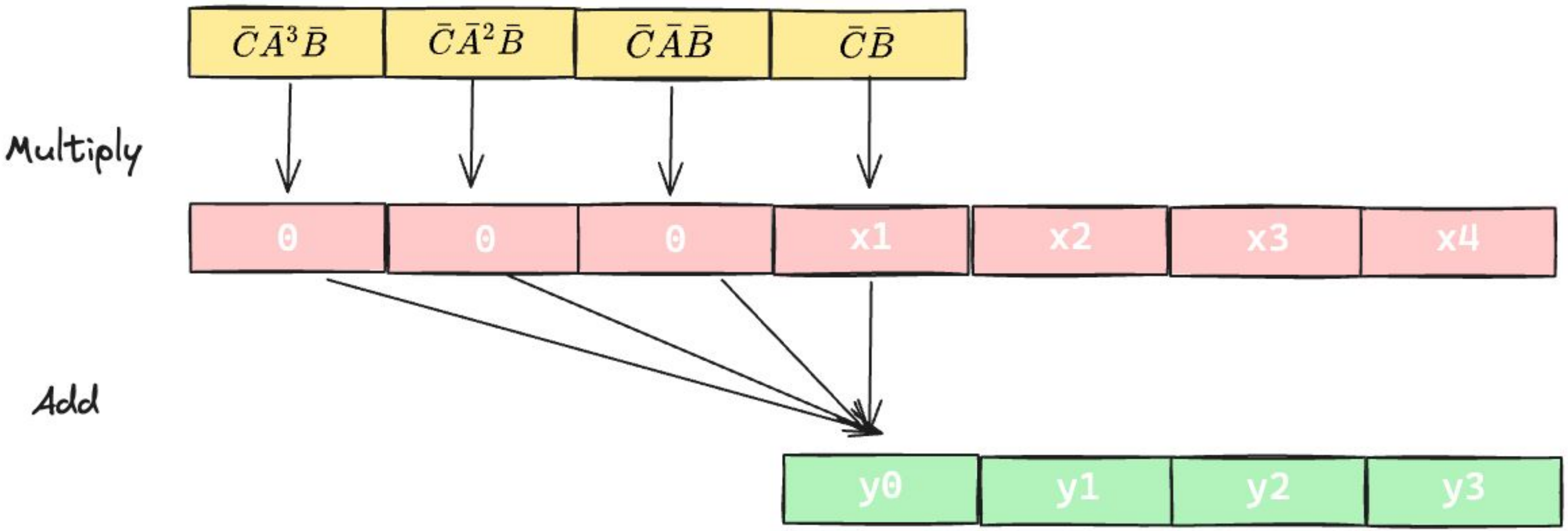
Extracting common coefficients, we get the SSM Kernel

$$\bar{K} = (\bar{C}\bar{A}^i\bar{B})_{i \in L}$$

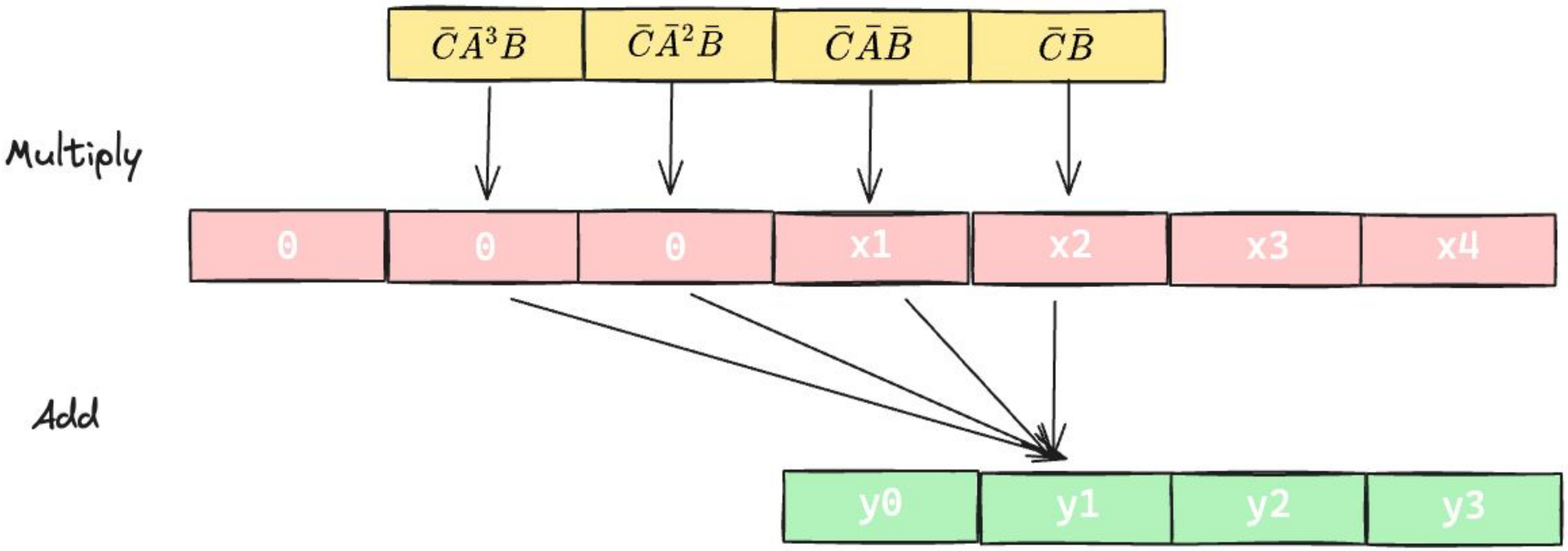


Source: Albert Gu

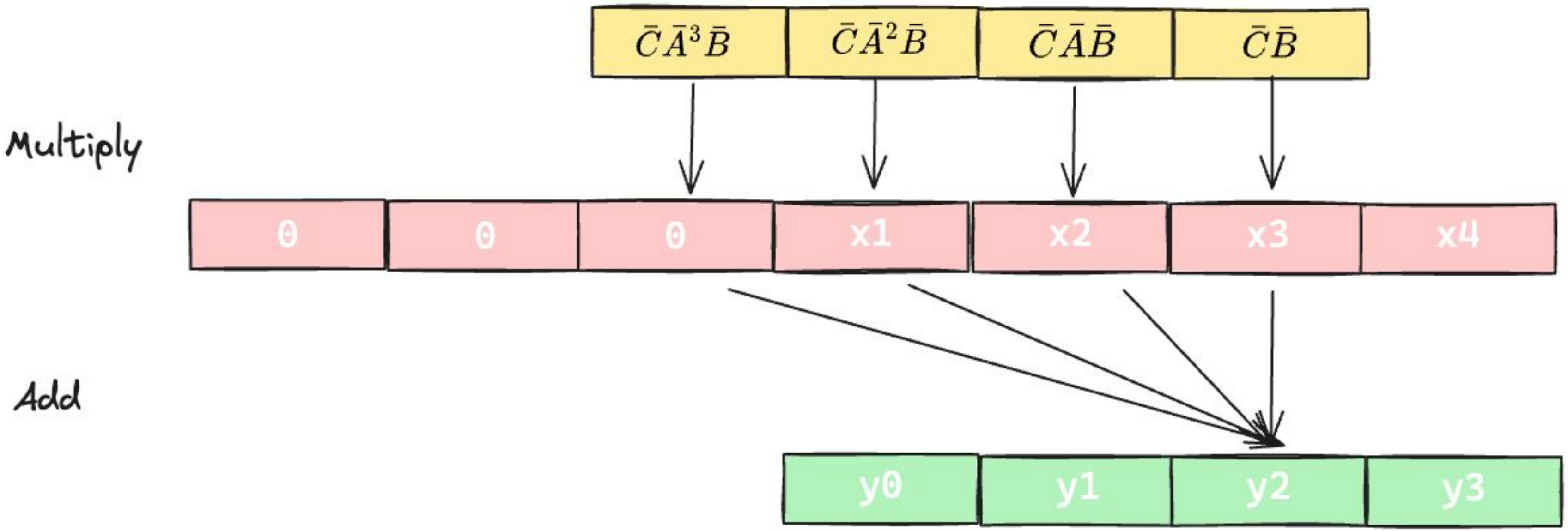
SSM: Convolutional Kernel



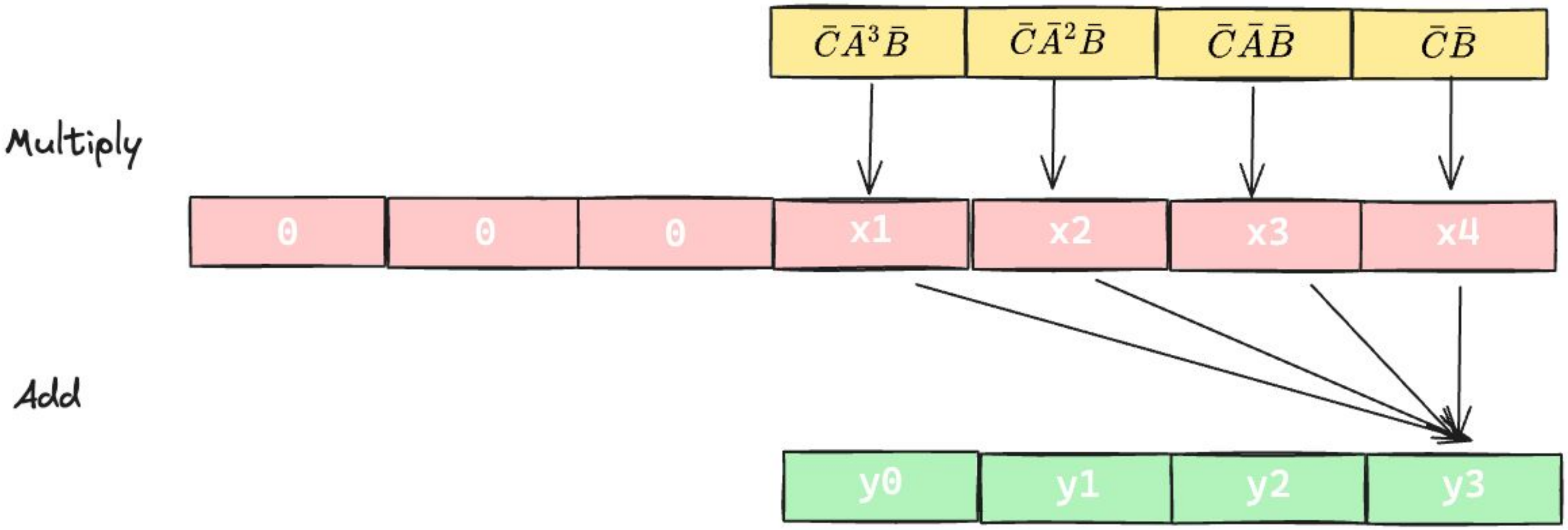
SSM: Convolutional Kernel



SSM: Convolutional Kernel



SSM: Convolutional Kernel



Deep SSM: Challenges

- **Modelling Challenge**
  - SSMs inherit problems of CNN,RNN on LRA
  - Random init A
    - **60%** acc sequential MNIST 🙁
- **Computation Challenge**
  - SSM has nice properties if $\bar{A}$ and $\bar{K}$ are known
  - Computing them is **hard**!
- **Computing the Kernel**
  - A power 🔼-> vanishing gradient?
  - A power 🔼-> O(D^2N) computation
    - Ideal -> O(N)

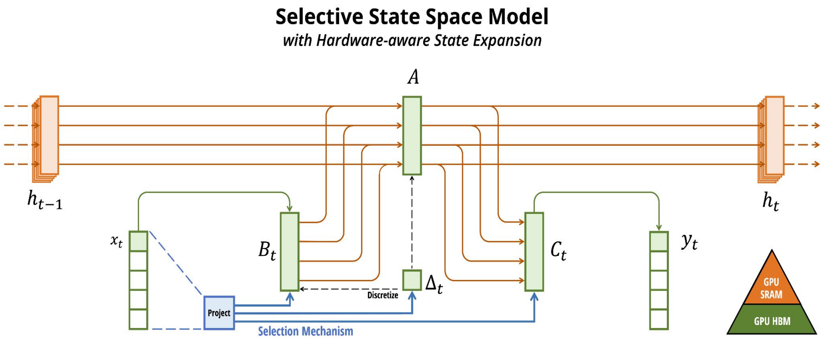
$$x_k = \bar{A}x_{k-1} + \bar{B}u_k$$

$$y_k = \bar{C}x_k + \bar{D}u_k$$

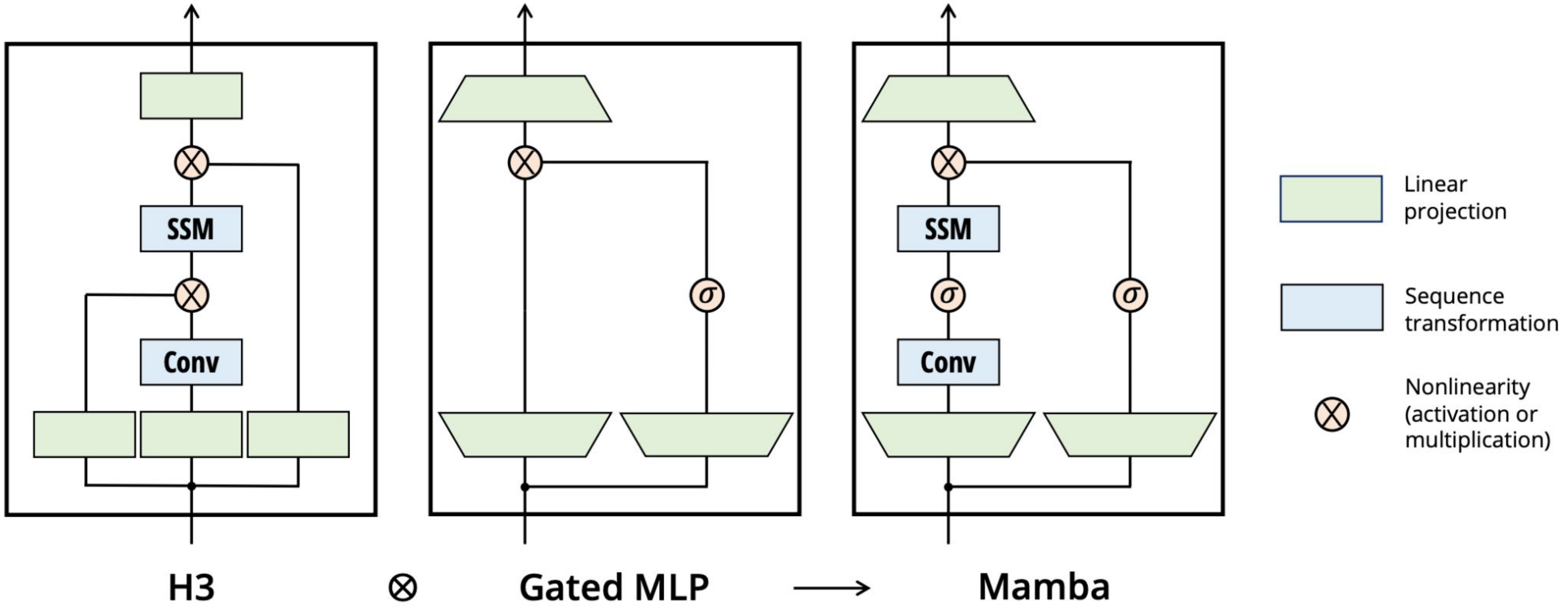
$$\bar{K} = (\bar{C}\bar{A}^i\bar{B})_{i \in L}$$

Mamba(S6): Selective SSM

- ➢ SSSSSS...🐍
- ➢ Improves SSM performance on copying tasks
- ➢ Handles input data that is varying in time
- ➢ Only supports recurrent form



Mamba: Architecture



Source: Mamba[Albert Gu, Tri Dao 2023]

Mamba: Implementation

Algorithm 1 SSM (S4)

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

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

Batch size: B
Prompt token size: L
Token dimensions: D

Algorithm 2 SSM + Selection (S6)

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

$\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\quad \triangleright$ Represents structured $N \times N$ matrix
$\boldsymbol{B} : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_B(x)$
$\boldsymbol{C} : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_C(x)$
$\Delta : (\mathtt{B}, \mathtt{L}, \mathtt{D}) \leftarrow \tau_\Delta(\mathsf{Parameter} + s_\Delta(x))$

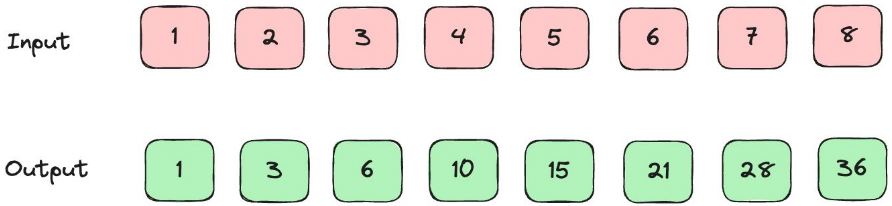
Linear layers

$\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathtt{B}, \mathtt{L}, \mathtt{D}, \mathtt{N}) \leftarrow \mathsf{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
$y \leftarrow \mathsf{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
$\quad \triangleright$ Time-varying: recurrence (*scan*) only
return y

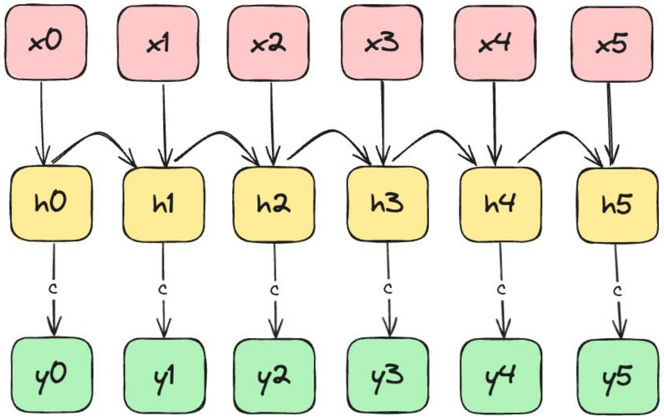
No parallel training 👎

Source: Mamba[Albert Gu, Tri Dao 2023]

Mamba: Scan Operation



Current val = Sum of previous val + input



$$x_t = \bar{A}x_{t-1} + \bar{B}u_t$$

$$y_t = Cx_t$$

Current State: Sum of previous state + input

References on Parallel Scan

- Slides from: Dan Grossman, U of Washington, http://homes.cs.washington.edu/~djg/teachingMaterials/spac
- Slides from David Walker, Princeton, https://www.cs.princeton.edu/courses/archive/fall13/cos326/lec/23-parallel-scan.pdf
- MIT 18.337 Modern Numerical Computing Lecture notes on Parallell Prefix: https://courses.csail.mit.edu/18.337/2004/book/Lecture_03-Parallel_Prefix.pdf
- Parallel Scan – Udacity, Slides from https://youtu.be/OO3o14cINbo?si=6ft0vTCU5FSu8Rl2
- Stanford CS149 Parallel Computing lecture: https://www.youtube.com/watch?v=Ba3TqxSgnTk

The Prefix-sum Problem

val prefix_sum : int array -> int array

`input`	6	4	16	10	16	14	2	8
`output`	6	10	26	36	52	66	68	76

The simple sequential algorithm: accumulate the sum from left to right

– Sequential algorithm: Work: $O(n)$, Span: $O(n)$
– Goal: a parallel algorithm with Work: $O(n)$, Span: O(log n)

Parallel Prefix Sum

The trick: *Use two passes*

- Each pass has $O(n)$ work and $O(\texttt{log}\ n)$ span
- So in total there is $O(n)$ work and $O(\texttt{log}\ n)$ span

First pass *builds a tree of sums bottom-up*

- the "up" pass

Second pass *traverses the tree top-down to compute prefixes*

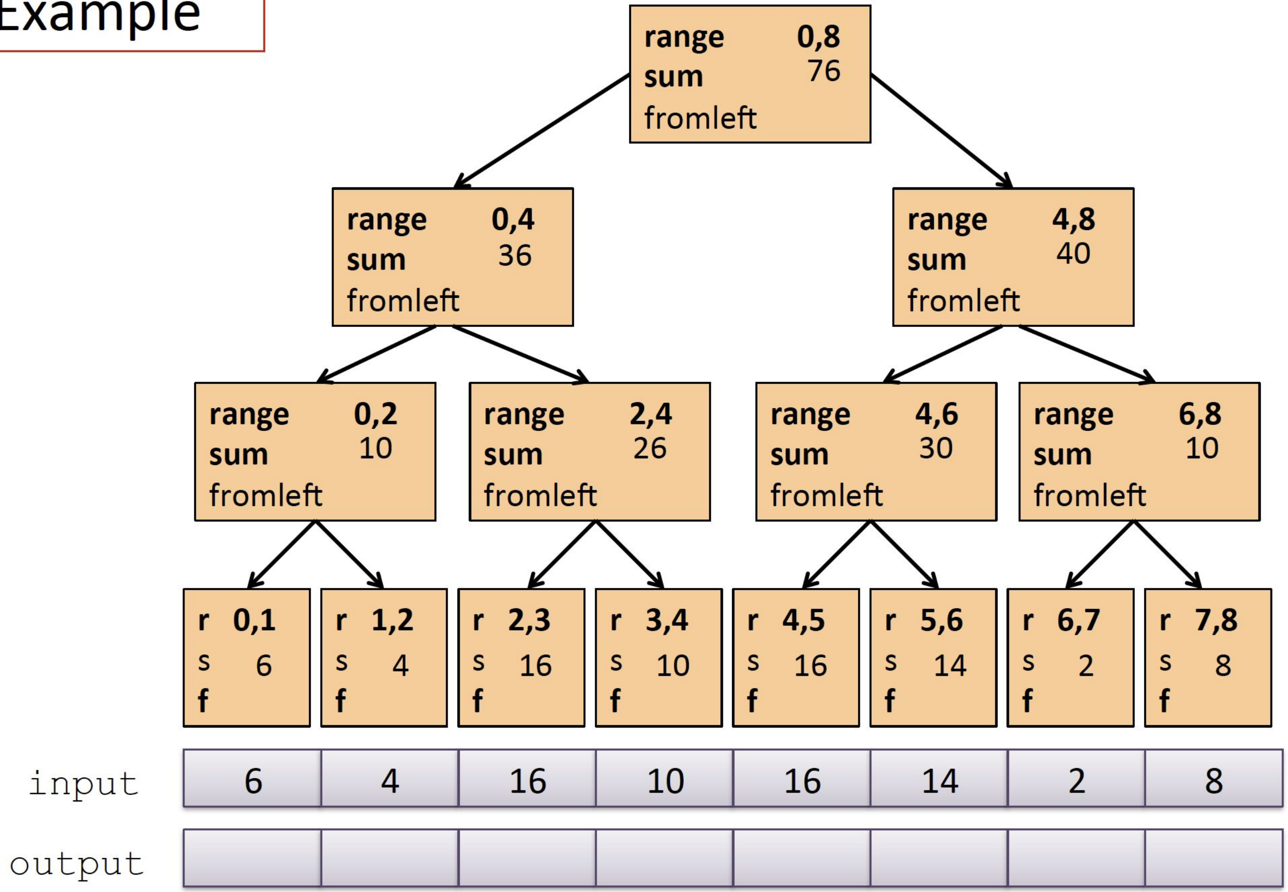
- the "down" pass

Historical note:

- Original algorithm due to R. Ladner and M. Fischer at the University of Washington in 1977

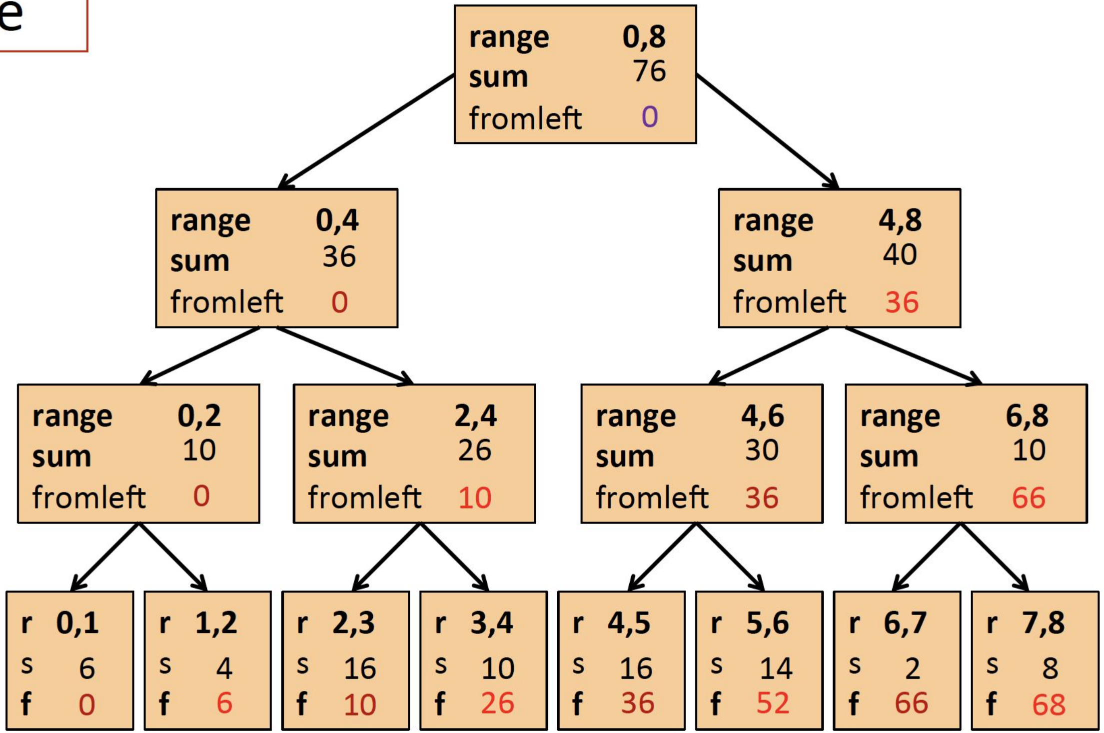
Parallel Prefix Sum

Example



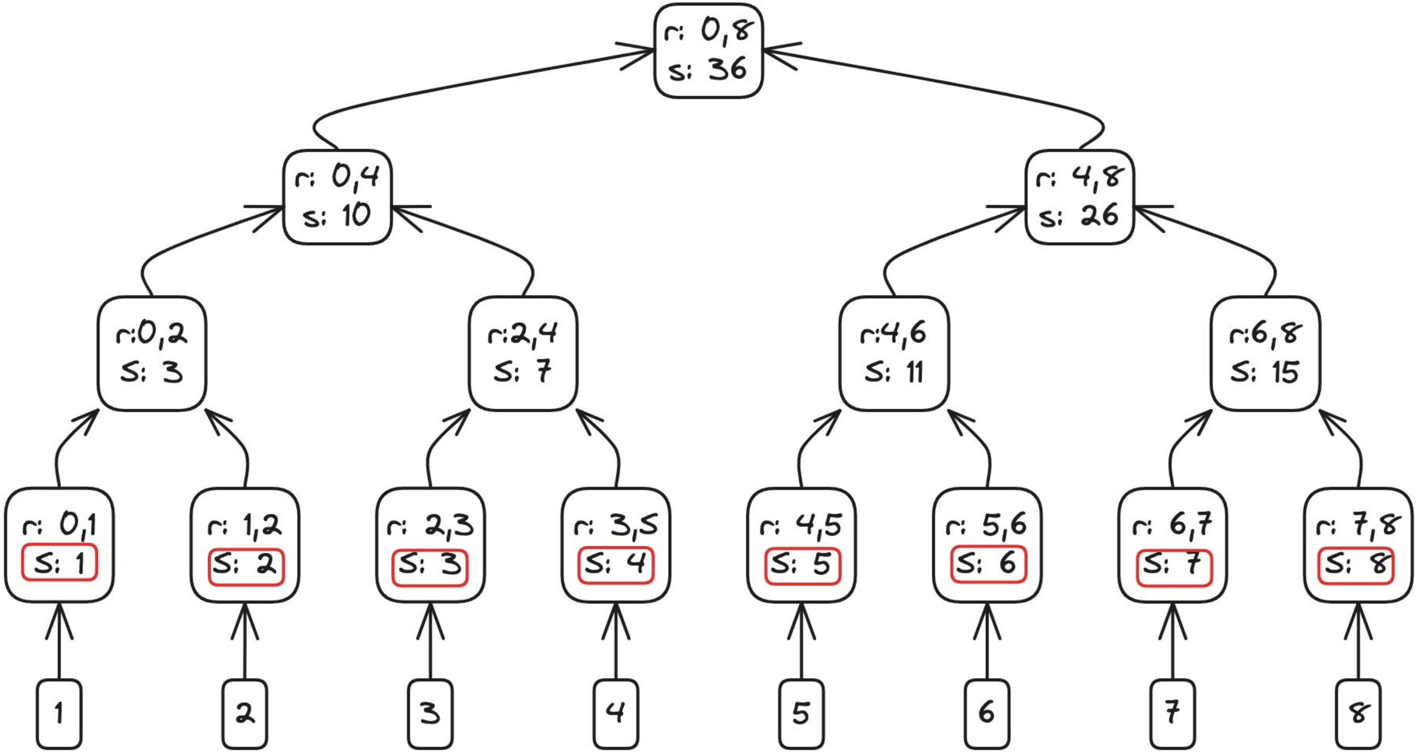
Parallel Prefix Sum

Example



input	6	4	16	10	16	14	2	8
output	6	10	26	36	52	66	68	76

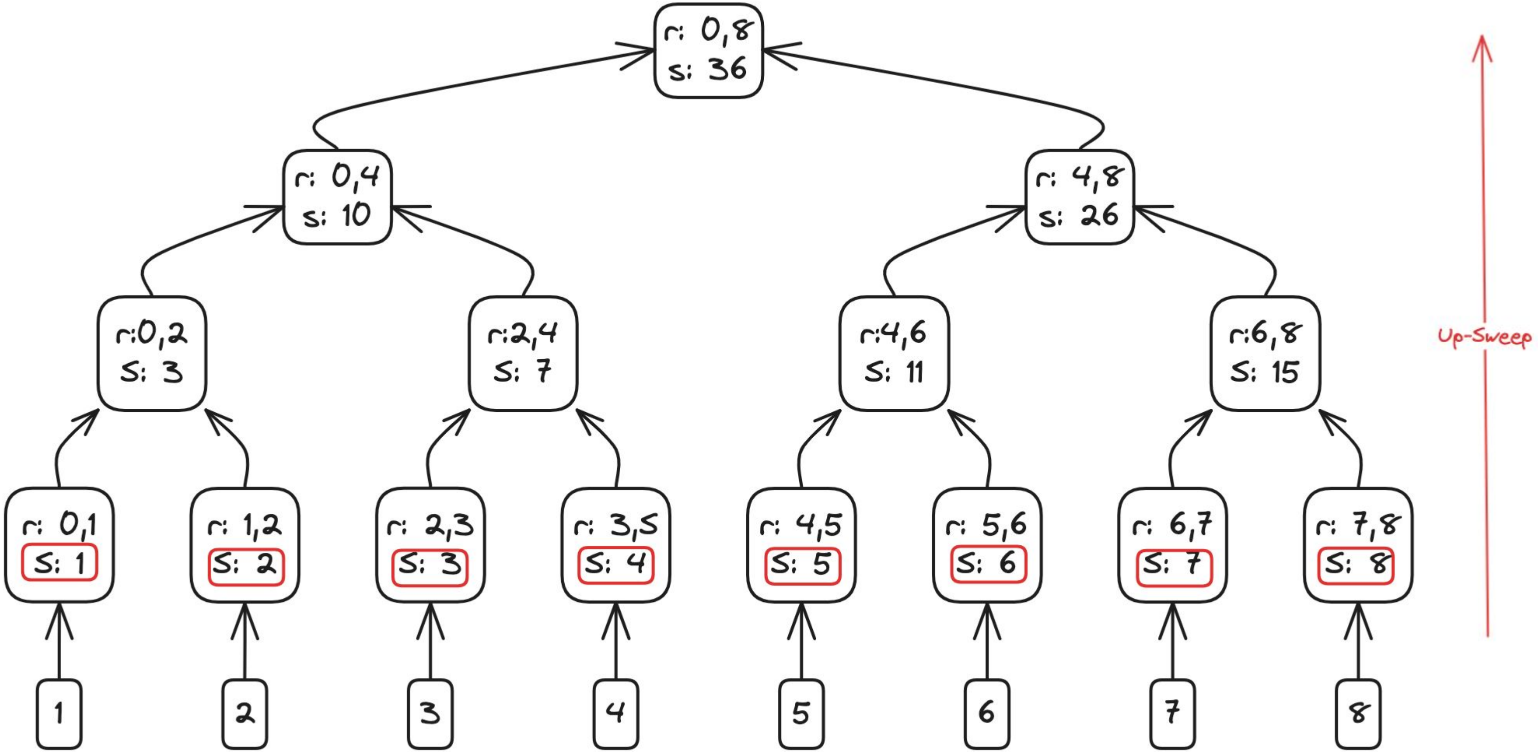
Parallel Scan: Another Example



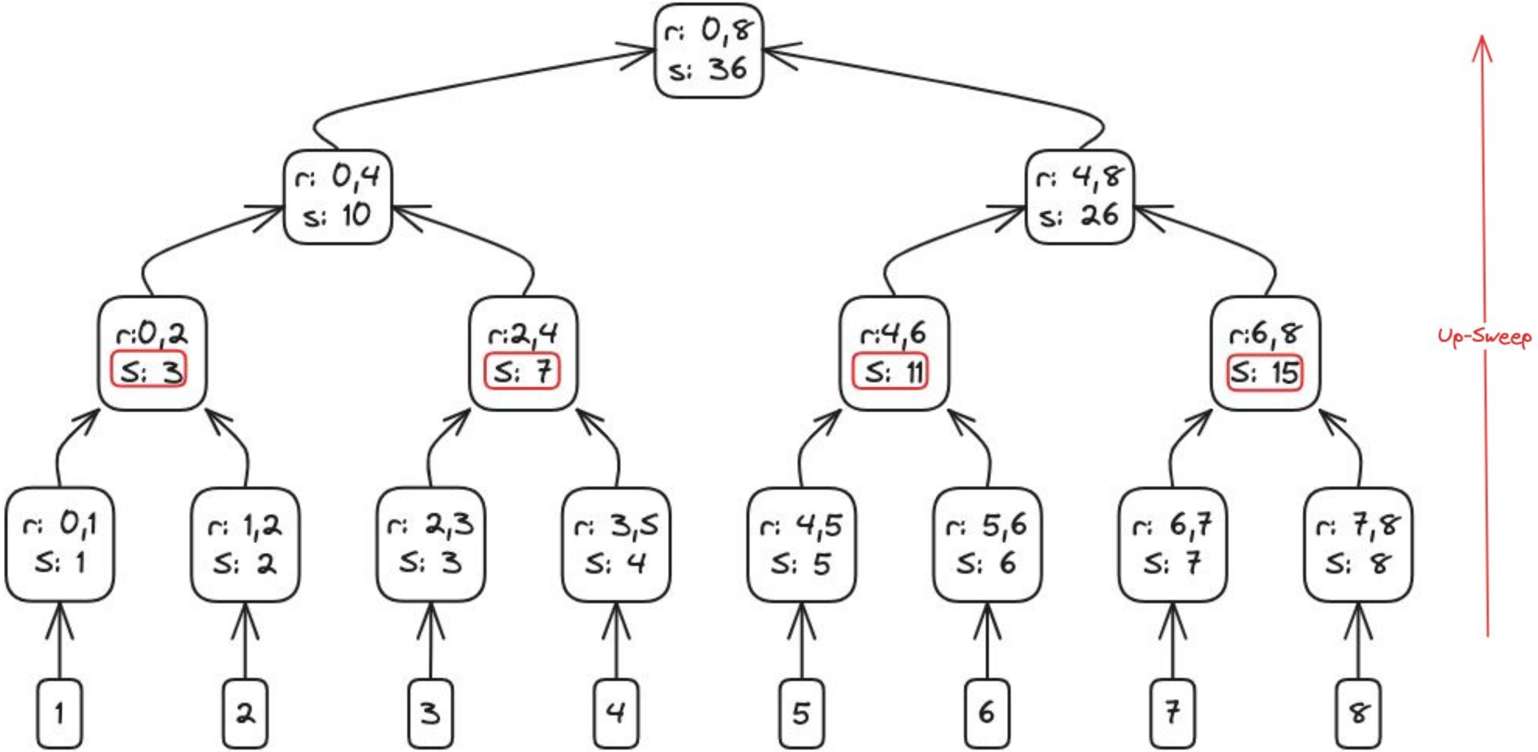
Parallelizing a Recurrence Relation Computation via Parallel Scan

- The parallel scan operation works in two passes:
    - An Up-sweep
    - A down-sweep
- During the up-sweep, we first break down our input into blocks. Here, assume that we use a block of 2 elements
- The variable ”R” just represents the range of elements being considered, and the variable “S” represent the sum
- Initialize the variable “S” for each left node in the tree as the element at that index
- Sum up child nodes and assign the value to “S” in the parent node

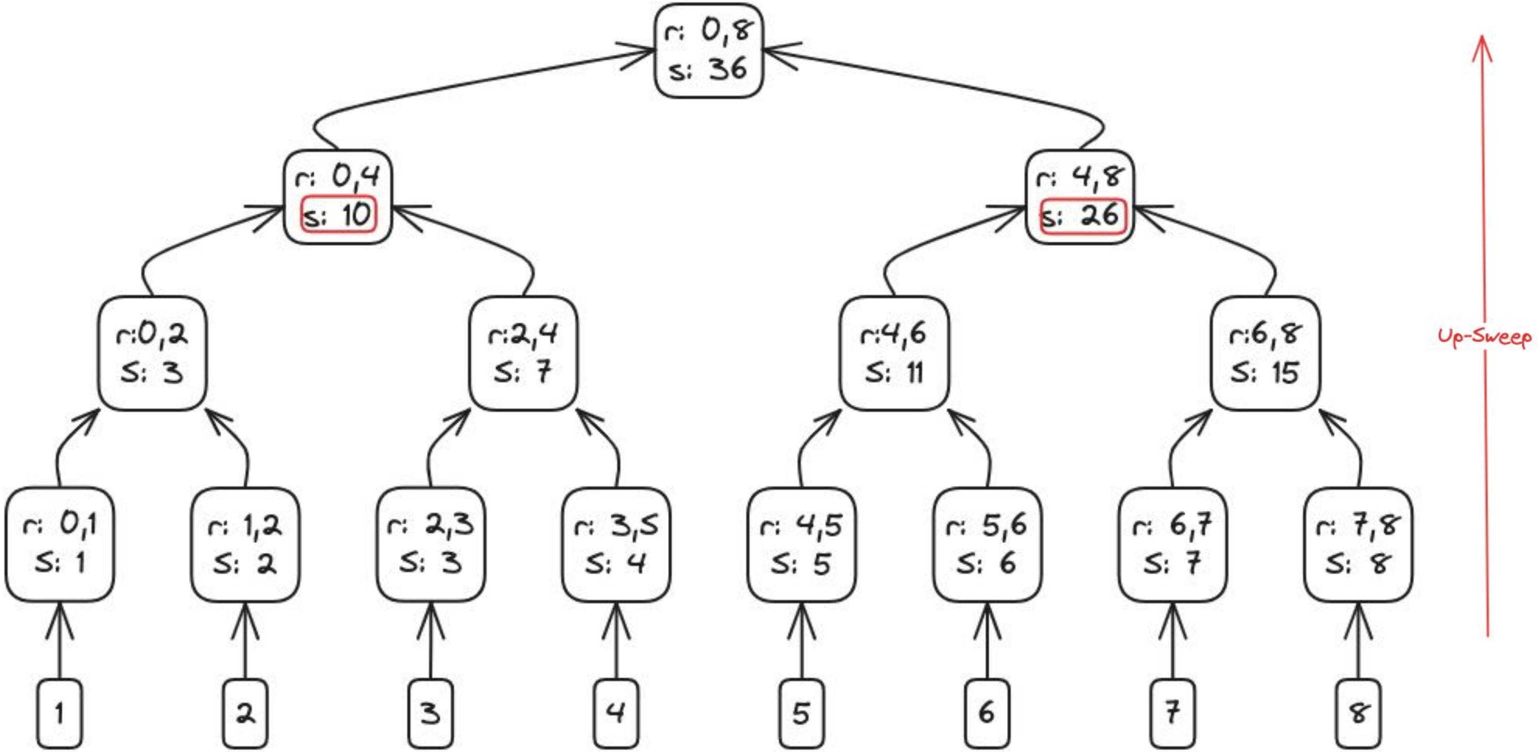
Parallel Scan: Up-Sweep



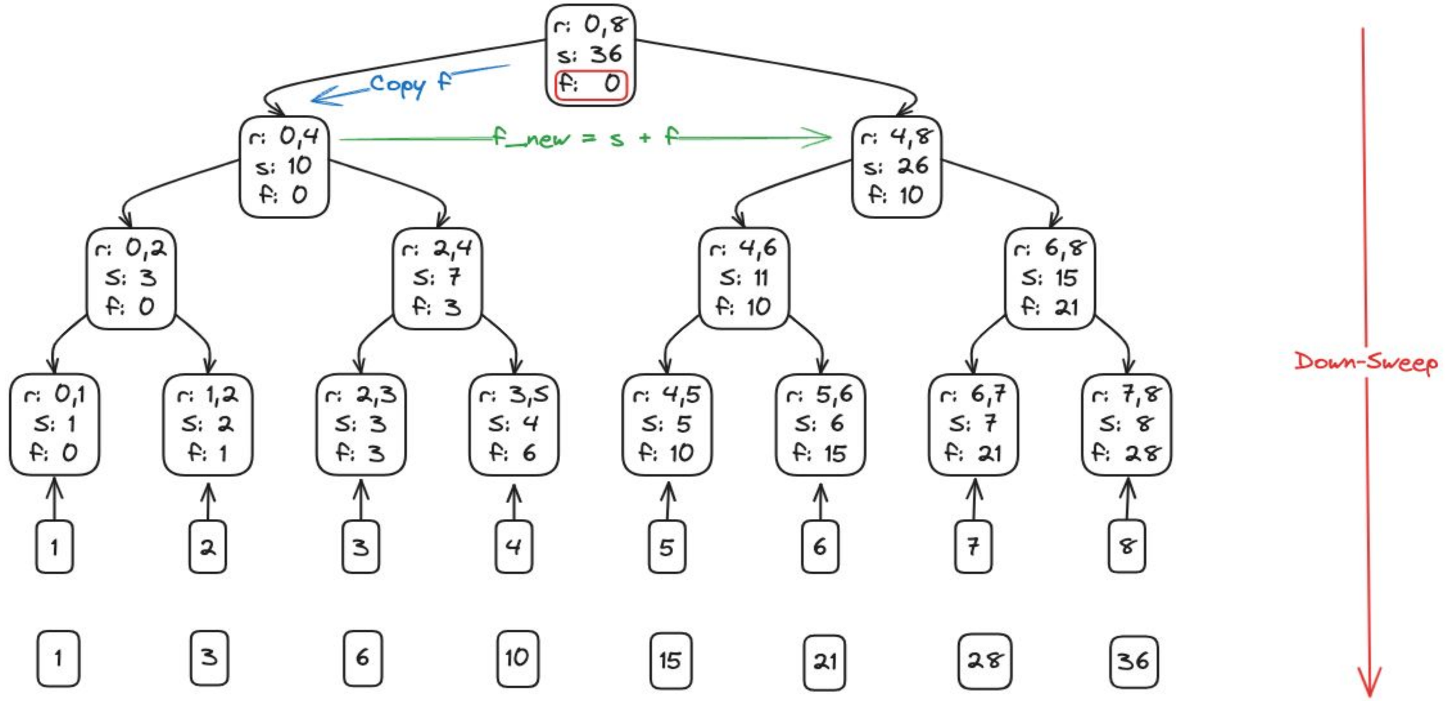
Parallel Scan: Up-Sweep



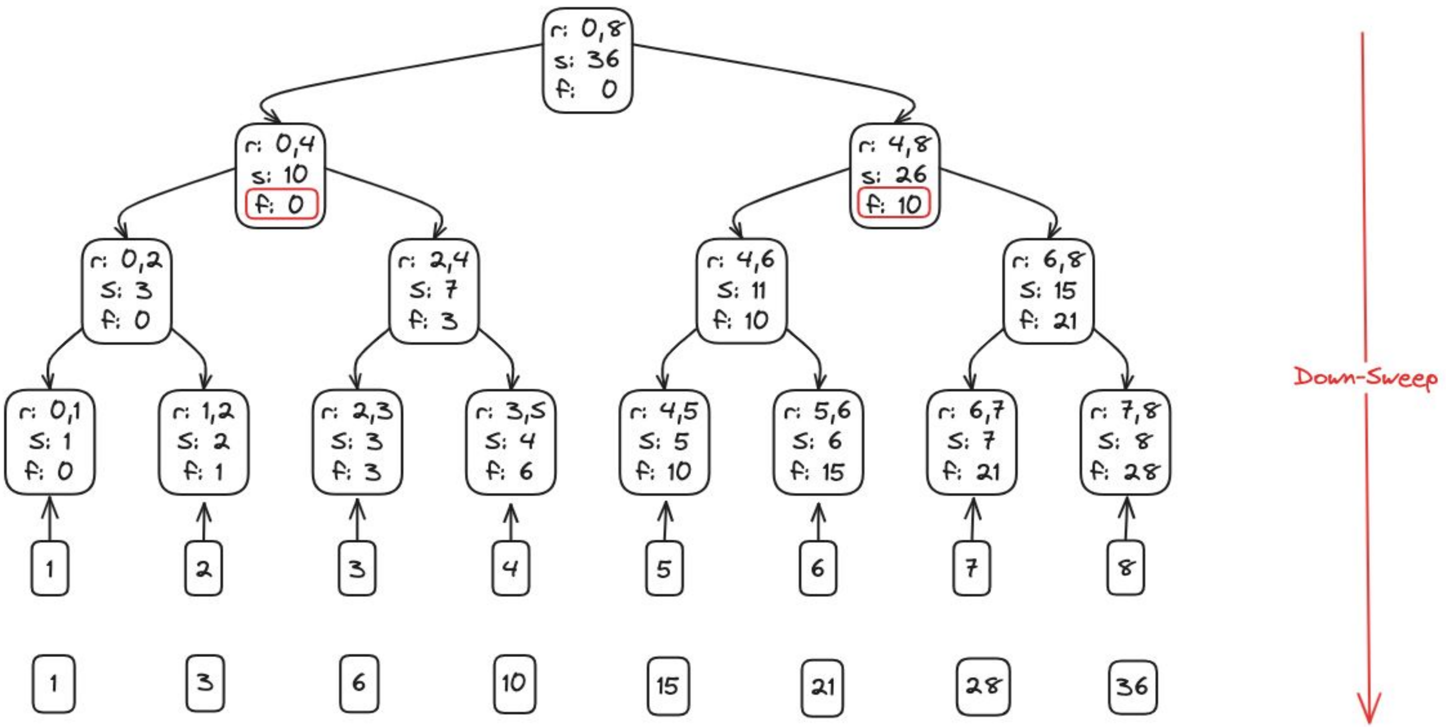
Parallel Scan: Up-Sweep



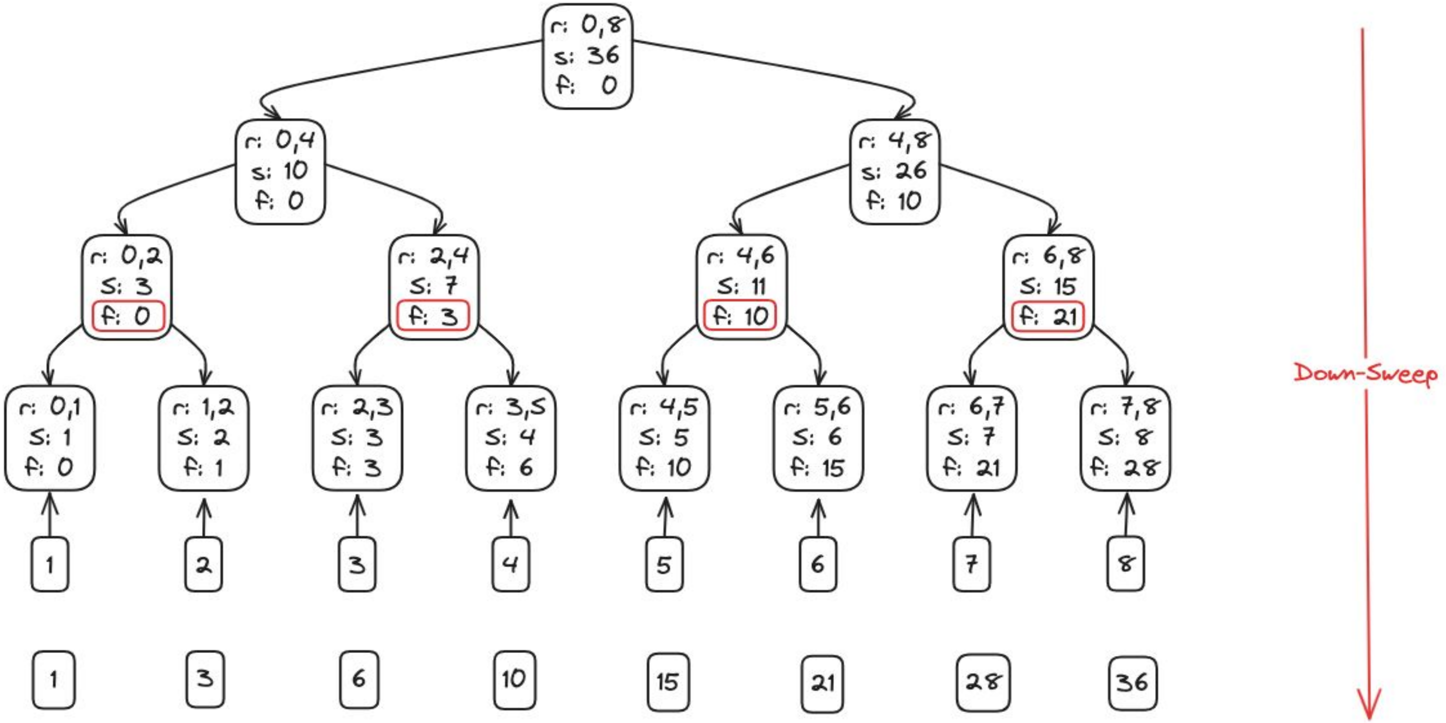
Parallel Scan: Down-Sweep



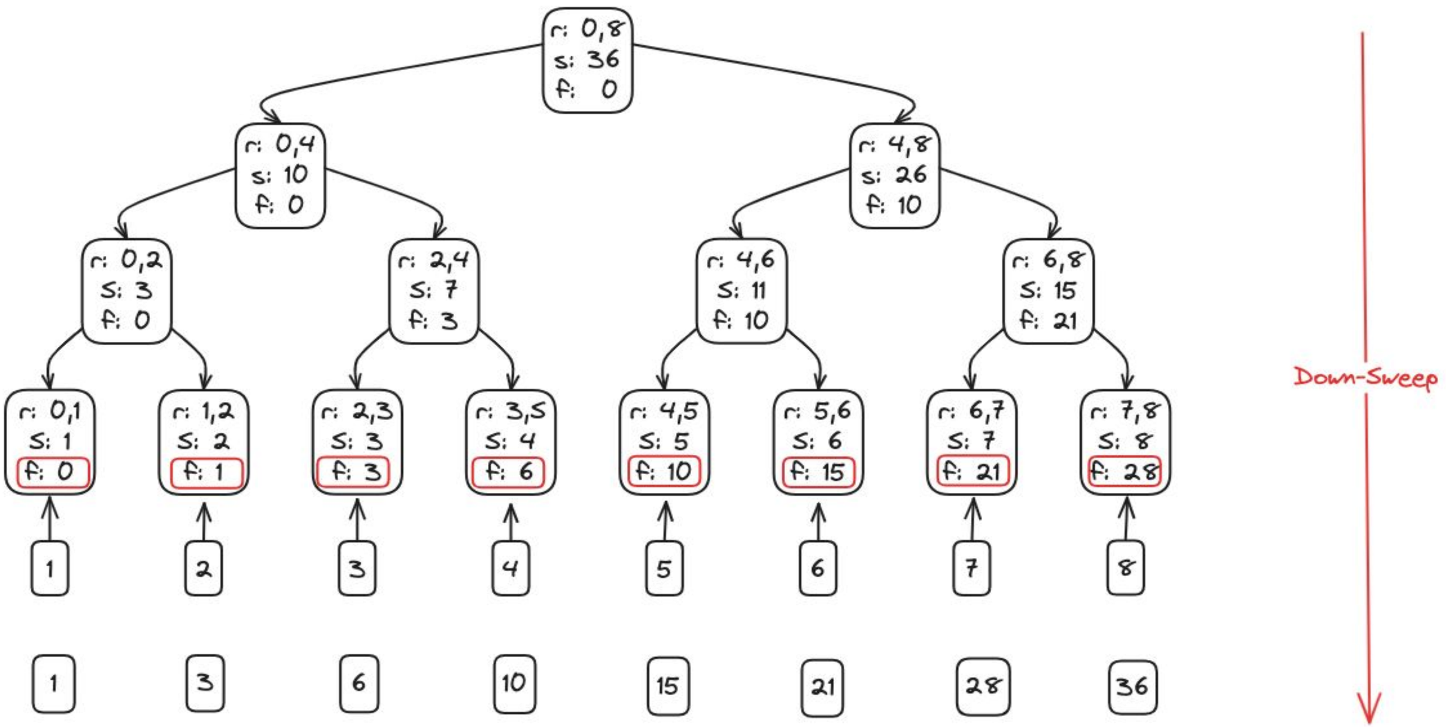
Parallel Scan: Down-Sweep



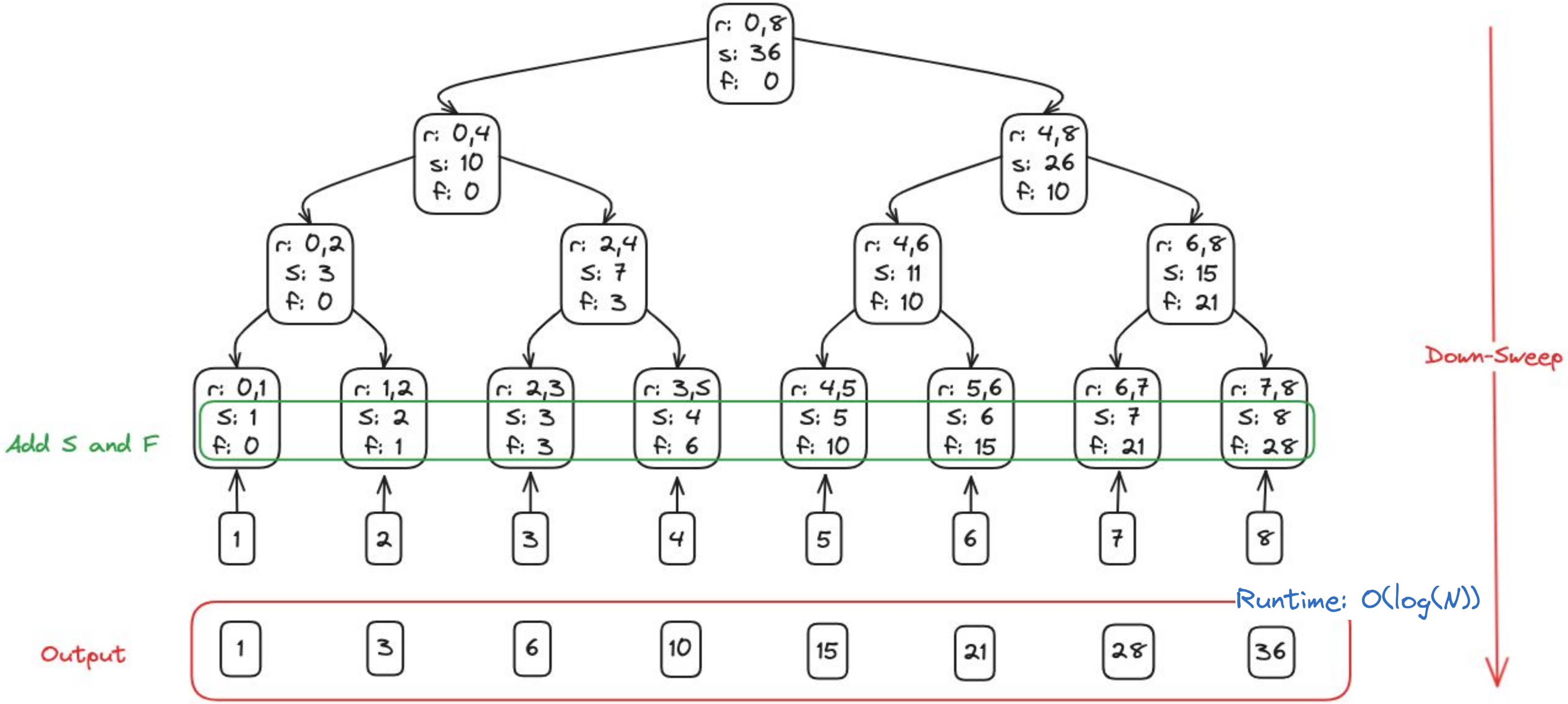
Parallel Scan: Down-Sweep



Parallel Scan: Down-Sweep



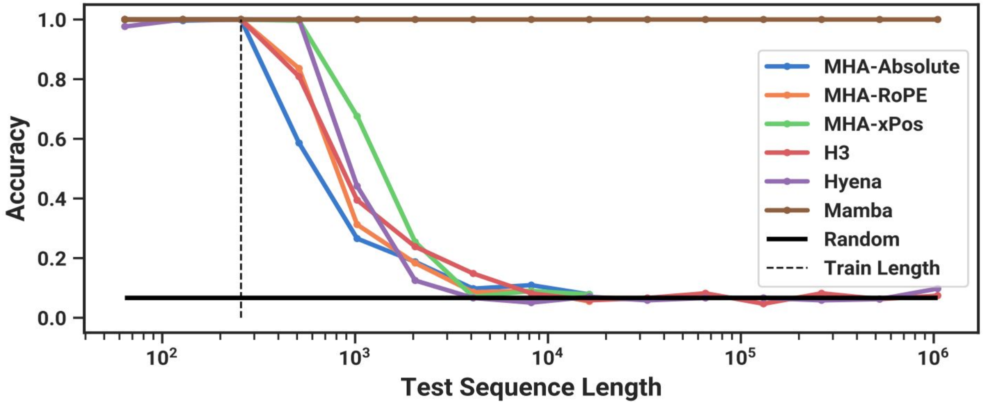
Parallel Scan: Final Result



Mamba: Synthetic Tasks Results

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**

Selective Copying



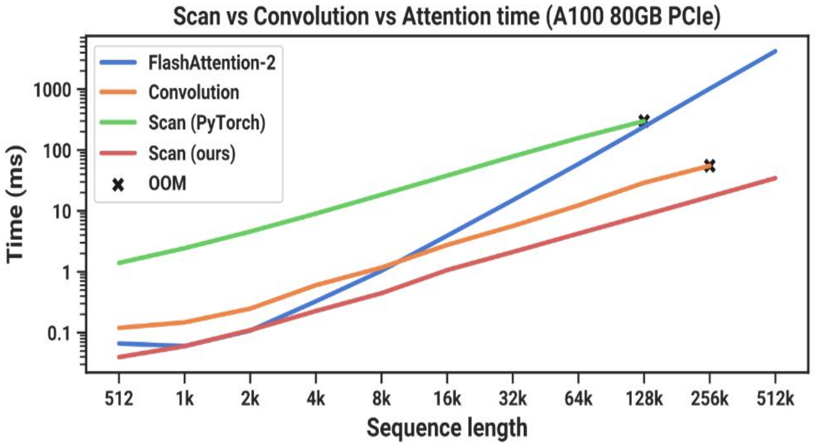
Induction Heads

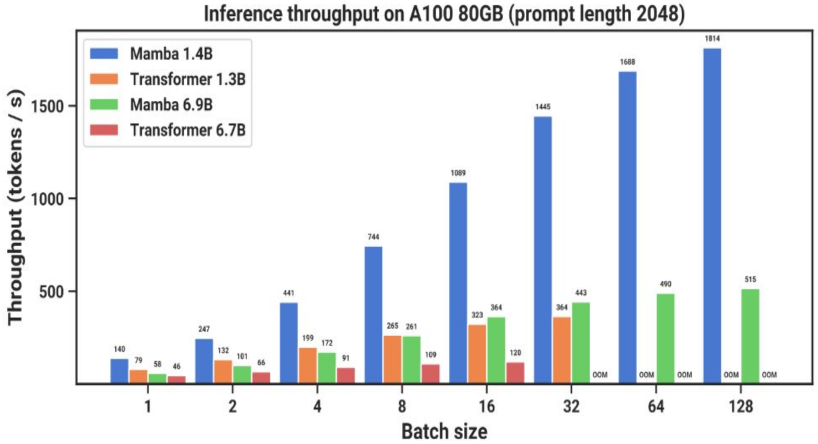
Mamba: LM Results

Model	Token.	Pile ppl ↓	LAMBADA ppl ↓	LAMBADA acc ↑	HellaSwag acc ↑	PIQA acc ↑	Arc-E acc ↑	Arc-C acc ↑	WinoGrande acc ↑	Average acc ↑
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**

Zero-shot Eval on popular benchmarks

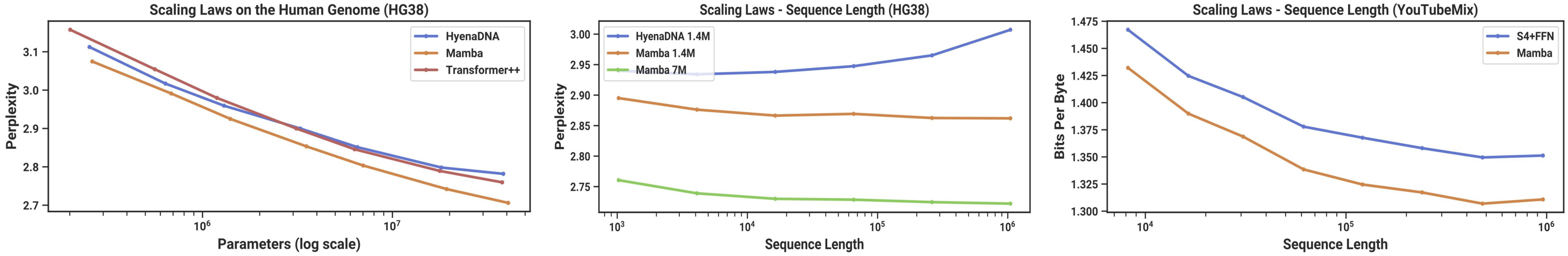
Mamba: Scaling Results





Source: Mamba[Albert Gu, Tri Dao 2023]

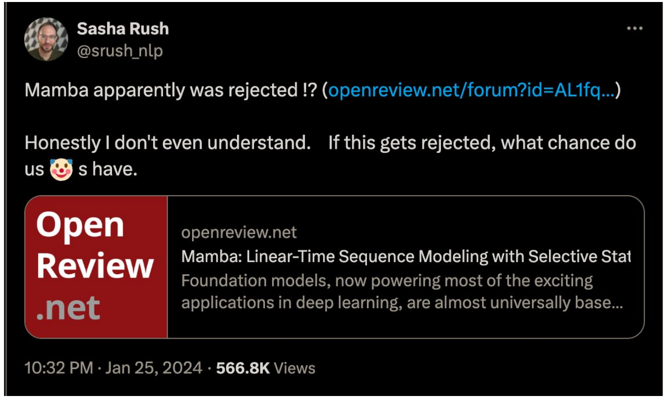
Mamba: Other Modalities



Zero-shot Eval on popular benchmarks

Mamba: Derivatives

- ➢ Jamba [O, Lieber et al., 2024]
- ➢ MambaByte [J Wang et al., 2024]
- ➢ Vision Mamba [Zhu, L, et al. , 2024]
- ➢ Mamba MoE [Pióro, Maciej, 2024]
- ➢ U-Mamba [Ma, Jun et al., 2024]
- ➢ Mamba-Morph [Guo, Tao et al., 2024]
- ➢ Swin-UMamba [Liu, J, et al., 2024]
- ➢ Graph Mamba [Behrouz et al., 2023]



Mamba rejected from ICLR 2024 😳

Mamba got
Outstanding Paper Award in COLM 2024

Theoretical Work on Capabilities of SSMs vs. Transformers

- Naoki Nishikawa, Taiji Suzuki. ”State Space Models are Provably Comparable to Transformers in Estimating Functions in Dynamic Token Selection,” to appear in ICLR 2025.
- Samy Jelassi, David Brandfonbrener, Sham M. Kakade, Eran Malach, “Repeat After Me: Transformers are Better than State Space Models at Copying, Transformers are Better than State Space Models at Copying”, ICML 2024.

Conclusion

- ➢ SSMs are a promising alternative to attention.
- ➢ Inference according to input was realized by Mamba.
- ➢ Hardware-Aware algorithms are the key to scale.
- ➢ Promising other research:
  - RWKV [Peng, Bo et al., 2023]
  - Gemini 1.5 [Gemini Team, 2024]

Moar SSM and Mamba Goodness 🐍

RWKV - Receptance Weighted Key Value

(pronounced as “RwaKuv”)

Receptance Weighted Key Value (RWKV): Reinventing RNNs for the Transformer Era

(based on the “old” RWKV-v4 architecture)

RWKV: Reinventing RNNs for the Transformer Era

Bo Peng[1*] Eric Alcaide[2,3,4*] Quentin Anthony[2,5*]
Alon Albalak[2,6] Samuel Arcadinho[2,7] Huanqi Cao[8] Xin Cheng[9] Michael Chung[10]
Matteo Grella[11] Kranthi Kiran GV[12] Xuzheng He[2] Haowen Hou[13] Przemysław Kazienko[14]
Jan Kocoń[14] Jiaming Kong[15] Bartłomiej Koptyra[14] Hayden Lau[2] Krishna Sri Ipsit Mantri[16]
Ferdinand Mom[17,18] Atsushi Saito[2,19] Xiangru Tang[20] Bolun Wang[27] Johan S. Wind[21] Stanisław Woźniak[14]
Ruichong Zhang[8] Zhenyuan Zhang[2] Qihang Zhao[22,23] Peng Zhou[27] Jian Zhu[24] Rui-Jie Zhu[25,26]

Presentation by Devan S. Zhun W. Ziyang W.

RWKV

Beyond Transformers - Intro to RWKV Architecture & The World Tokenizer
Eugene Cheah & Harrison Vanderbyl, Dec 2023
https://www.youtube.com/watch?v=I-HMKky7Qsw

Performance of Eagle-7B, an RWKV-v5 Architecture, Jan 2024
https://www.youtube.com/watch?v=gHdRgfmAVIw

Latest model as of Feb 2025: RWKV-v7 (Goose)
https://github.com/BlinkDL/RWKV-LM/blob/main/RWKV-v7/rwkv_v7_demo.py

High-Level Overview of RWKV

RNN vs. CNN vs. Quai-RNN

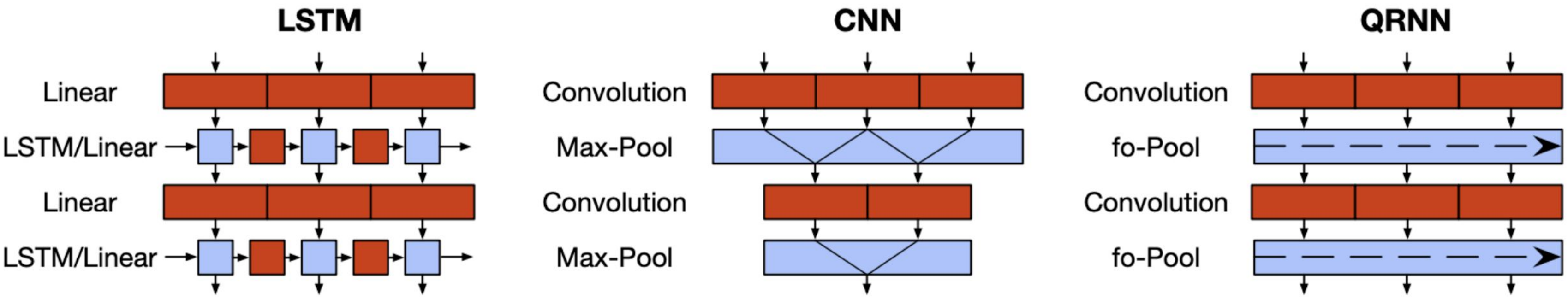
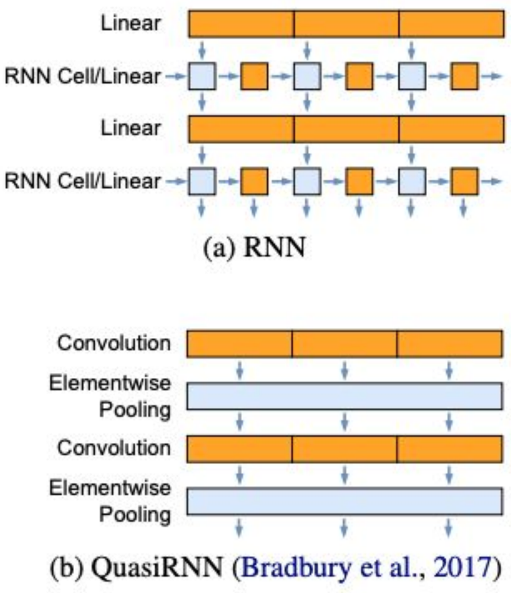


Figure 1: Block diagrams showing the computation structure of the QRNN compared with typical LSTM and CNN architectures. Red signifies convolutions or matrix multiplications; a continuous block means that those computations can proceed in parallel. Blue signifies parameterless functions that operate in parallel along the channel/feature dimension. LSTMs can be factored into (red) linear blocks and (blue) elementwise blocks, but computation at each timestep still depends on the results from the previous timestep.

Source: J. Bradbury et al, Quasi RNN, 2016, https://arxiv.org/pdf/1611.01576v2 See also: https://paperswithcode.com/method/qrnn

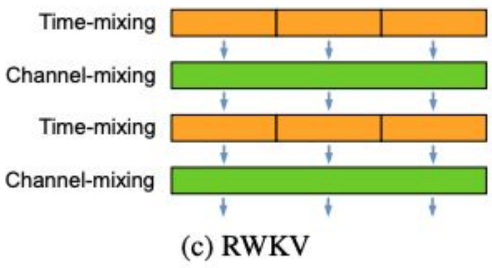
From RNN to Quai-RNN to RWKV

RWKV builds on existing sequence-to-sequence models with an architecture of stacked residual blocks (alternating time-mixing and channel-mixing blocks)



(a) RNN

(b) QuasiRNN (Bradbury et al., 2017)

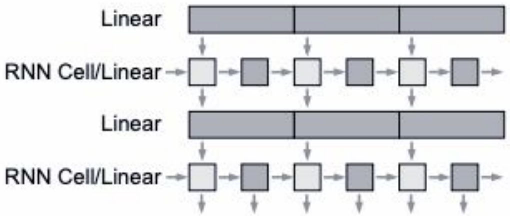


(c) RWKV

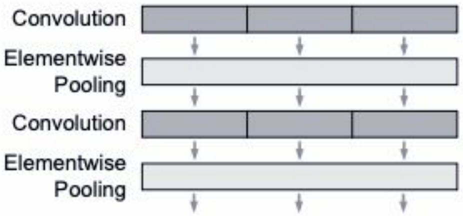
(Continuous blocks can occur simultaneously)

From RNN to Quai-RNN to RWKV

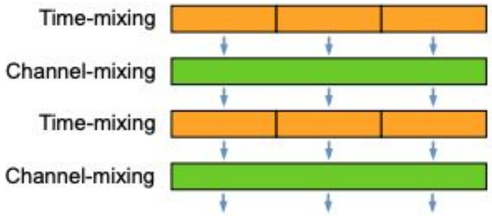
RWKV builds on existing sequence-to-sequence models with an architecture of stacked residual blocks (alternating time-mixing and channel-mixing blocks)



(a) RNN

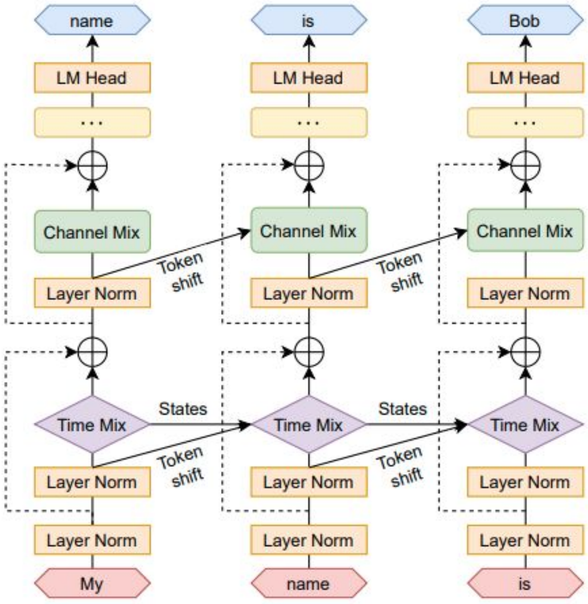


(b) QuasiRNN (Bradbury et al., 2017)



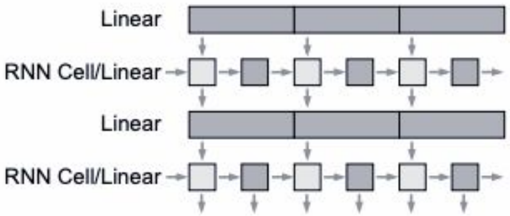
(c) RWKV

(Continuous blocks can occur simultaneously)

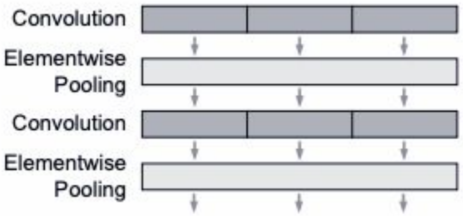


From RNN to Quai-RNN to RWKV

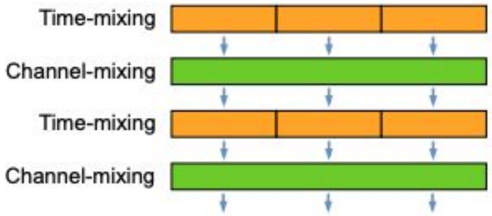
RWKV builds on existing sequence-to-sequence models with an architecture of stacked residual blocks (alternating time-mixing and channel-mixing blocks)



(a) RNN

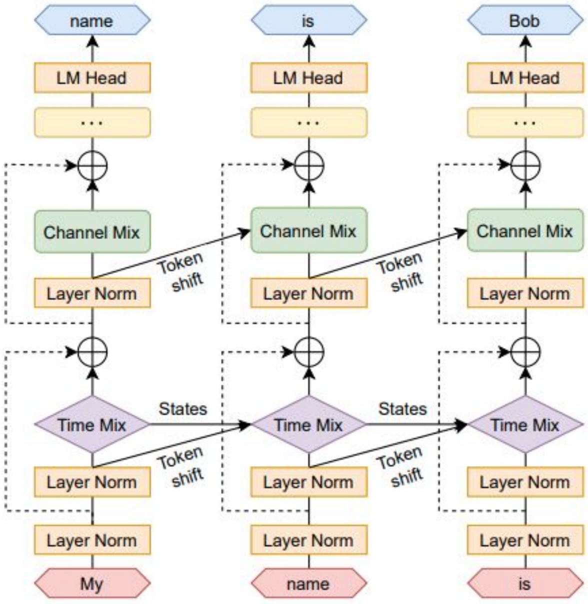


(b) QuasiRNN (Bradbury et al., 2017)



(c) RWKV

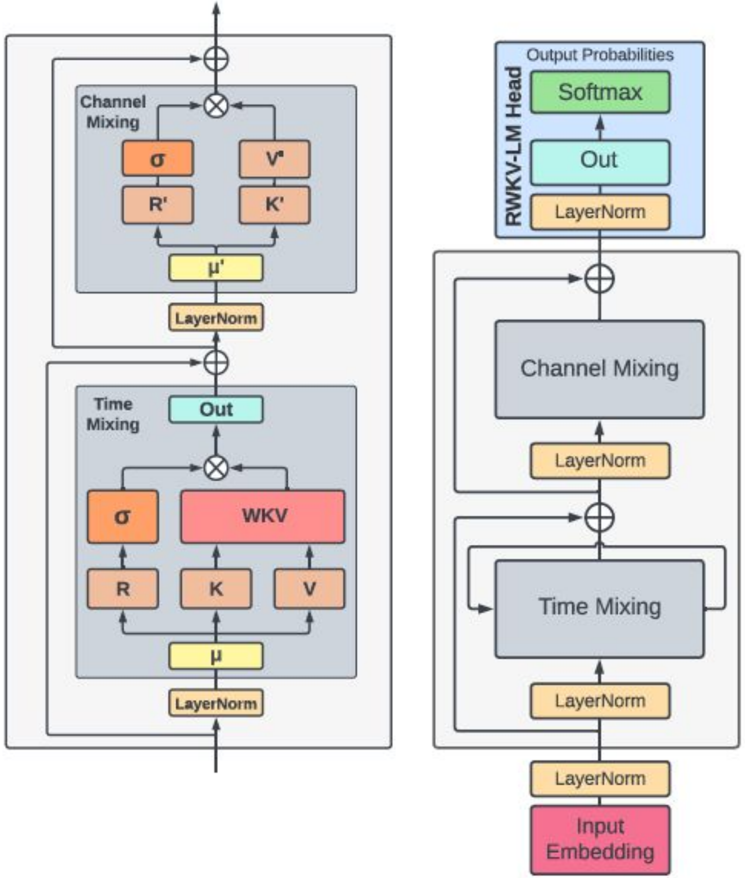
(Continuous blocks can occur simultaneously)



(Receptive field grows in similar manner as CNN)

RWKV: architecture overview

- R: **Receptance** vector acting as the acceptance of past information.
- W: **Weight** is the (trainable) positional weight decay vector.
- K: **Key**, analogous to K in standard attention.
- V: **Value**, analogous to V in standard attention.



Defining RWKV

The RWKV architecture derives its name from the four primary model elements used in the time-mixing and channel-mixing blocks:

- R: **Receptance** vector acting as the acceptance of past information
- W: **Weight** is the positional weight decay vector (trainable model parameter)
- K: **Key** is a vector analogous to K in traditional attention
- V : **Value** is a vector analogous to V in traditional attention

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}(QK^\top)V,$$

$$\mathrm{Attn}(Q, K, V)_t = \frac{\sum_{i=1}^{T} e^{q_t^\top k_i} v_i}{\sum_{i=1}^{T} e^{q_t^\top k_i}}$$

$$\mathrm{Attn}^+(W, K, V)_t = \frac{\sum_{i=1}^{t} e^{w_{t,i}+k_i} v_i}{\sum_{i=1}^{t} e^{w_{t,i}+k_i}}$$

$$w_{t,i} = -(t-i)w,$$

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

Defining RWKV

The RWKV architecture derives its name from the four primary model elements used in the time-mixing and channel-mixing blocks:

- R: **Receptance** vector acting as the acceptance of past information
- W: **Weight** is the positional weight decay vector (trainable model parameter)
- K: **Key** is a vector analogous to K in traditional attention
- V : **Value** is a vector analogous to V in traditional attention

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}(QK^\top)V,$$

$$\mathrm{Attn}(Q, K, V)_t = \frac{\sum_{i=1}^{T} e^{q_t^\top k_i} v_i}{\sum_{i=1}^{T} e^{q_t^\top k_i}}$$

wkv$_t$ acts in place of Attn(Q, K, V) without quadratic cost (note the scalar interactions)

$$\mathrm{Attn}^+(W, K, V)_t = \frac{\sum_{i=1}^{t} e^{w_{t,i}+k_i} v_i}{\sum_{i=1}^{t} e^{w_{t,i}+k_i}}$$

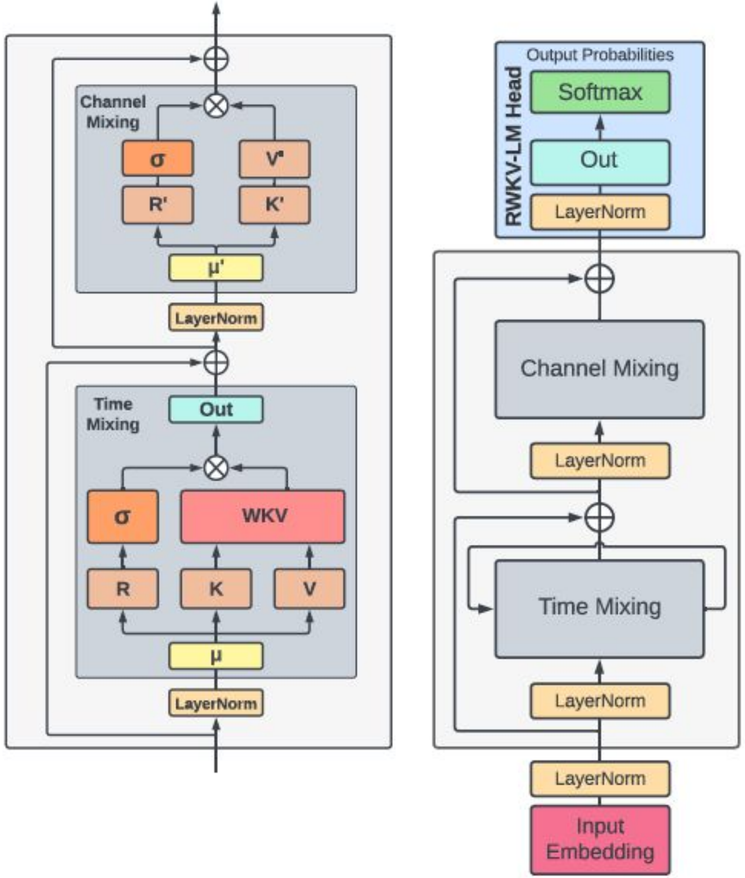
$$\boxed{w_{t,i} = -(t-i)w,}$$

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

”Inspired” by “An Attention-Free Transformer (AFT)”, by Zhai et al, 2021

RWKV: architecture overview

- R: **Receptance** vector acting as the acceptance of past information.
- W: **Weight** is the (trainable) positional weight decay vector.
- K: **Key**, analogous to K in standard attention.
- V: **Value**, analogous to V in standard attention.

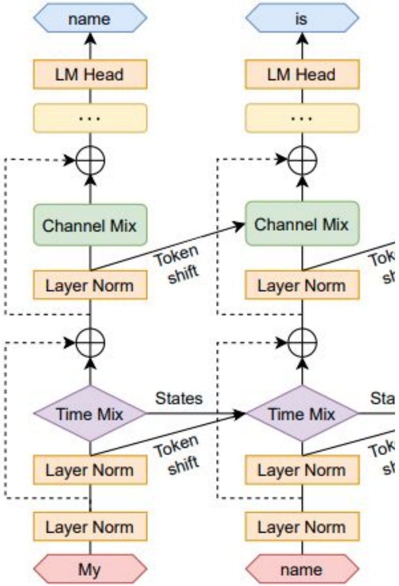


Channel and Time Mixing

Channel Mixing

Analog: shorter-term memory
(Mix over channels only)

$$r_t = W_r \cdot (\mu_r x_t + (1-\mu_r)x_{t-1}),$$
$$k_t = W_k \cdot (\mu_k x_t + (1-\mu_k)x_{t-1}),$$
$$o_t = \sigma(r_t) \odot (W_v \cdot \max(k_t, 0)^2),$$



Time Mixing

Analog: longer-term memory
(Mix over time *and* channels)

$$r_t = W_r \cdot (\mu_r x_t + (1-\mu_r)x_{t-1}),$$
$$k_t = W_k \cdot (\mu_k x_t + (1-\mu_k)x_{t-1}),$$
$$v_t = W_v \cdot (\mu_v x_t + (1-\mu_v)x_{t-1}),$$
$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}},$$
$$o_t = W_o \cdot (\sigma(r_t) \odot wkv_t),$$

Please refer to https://wiki.rwkv.com/advance/architecture.html for more thorough analysis from the authors about the short term and long term natures of channel and time mixing.

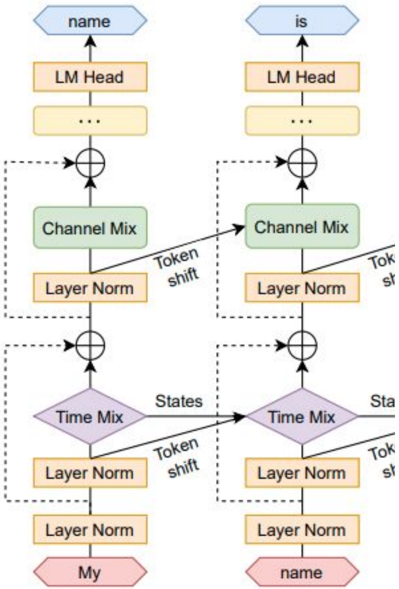
Channel and Time Mixing

Channel Mixing

Analog: shorter-term memory
(Mix over channels only)

$$r_t = W_r \cdot (\mu_r x_t + (1-\mu_r) x_{t-1}),$$
$$k_t = W_k \cdot (\mu_k x_t + (1-\mu_k) x_{t-1}),$$
$$o_t = \sigma(r_t) \odot (W_v \cdot \max(k_t, 0)^2),$$

"forget gate"
(sigmoid of receptance)



Time Mixing

Analog: longer-term memory
(Mix over time *and* channels)

$$r_t = W_r \cdot (\mu_r x_t + (1-\mu_r) x_{t-1}),$$
$$k_t = W_k \cdot (\mu_k x_t + (1-\mu_k) x_{t-1}),$$
$$v_t = W_v \cdot (\mu_v x_t + (1-\mu_v) x_{t-1}),$$
$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}},$$
$$o_t = W_o \cdot (\sigma(r_t) \odot wkv_t),$$

"forget gate"

Please refer to https://wiki.rwkv.com/advance/architecture.html for more thorough analysis from the authors about the short term and long term natures of channel and time mixing.

Channel and Time Mixing

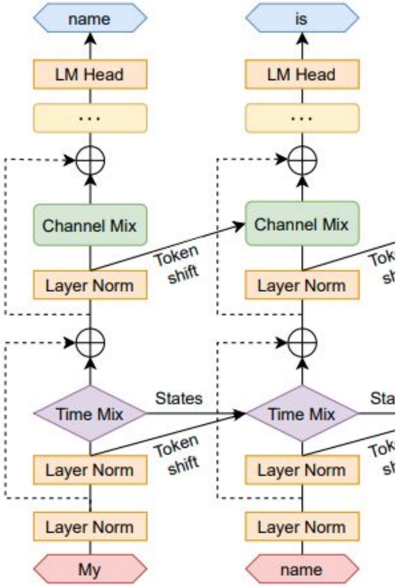
Channel Mixing

Analog: shorter-term memory
(Mix over channels only)

(token shifts can just be considered as x̃)

$$r_t = W_r \cdot (\mu_r x_t + (1-\mu_r)x_{t-1}),$$
$$k_t = W_k \cdot (\mu_k x_t + (1-\mu_k)x_{t-1}),$$
$$o_t = \sigma(r_t) \odot (W_v \cdot \max(k_t, 0)^2),$$

"forget gate"
(sigmoid of receptance)



Time Mixing

Analog: longer-term memory
(Mix over time *and* channels)

(token shifts can just be considered as x̃)

$$r_t = W_r \cdot (\mu_r x_t + (1-\mu_r)x_{t-1}),$$
$$k_t = W_k \cdot (\mu_k x_t + (1-\mu_k)x_{t-1}),$$
$$v_t = W_v \cdot (\mu_v x_t + (1-\mu_v)x_{t-1}),$$
$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}},$$
$$o_t = W_o \cdot (\sigma(r_t) \odot wkv_t),$$

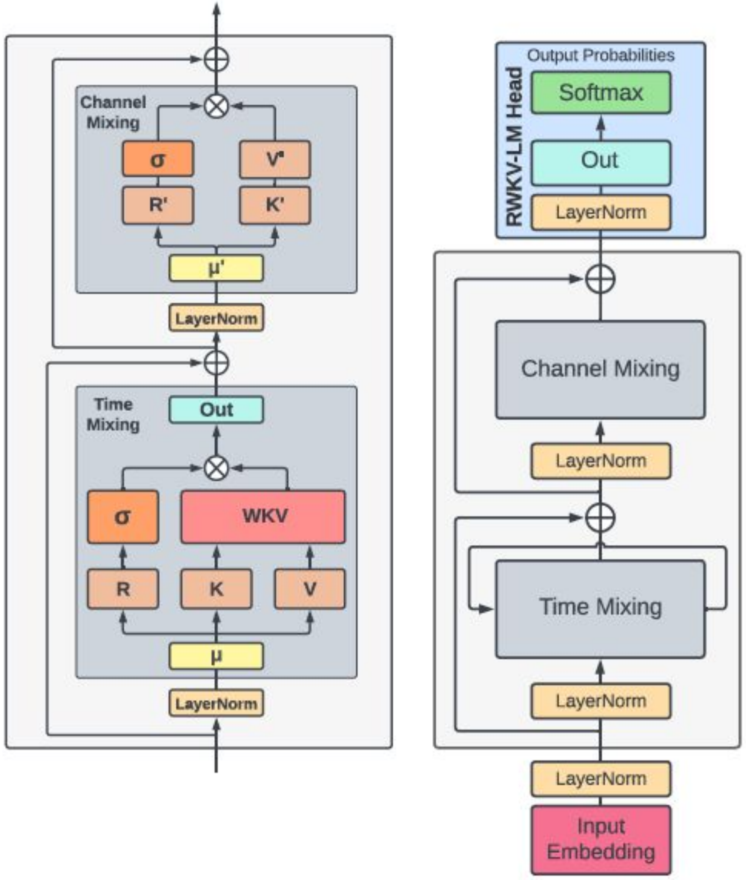
"forget gate"

Please refer to https://wiki.rwkv.com/advance/architecture.html for more thorough analysis from the authors about the short term and long term natures of channel and time mixing.

RWKV In-Depth

RWKV: architecture overview

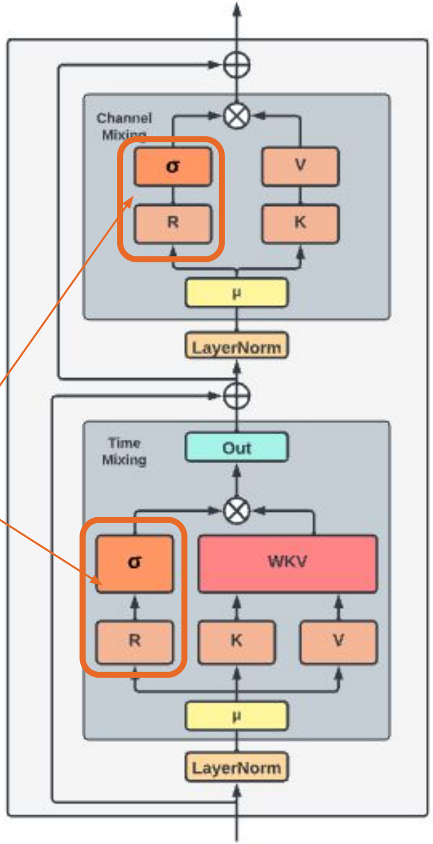
- R: **Receptance** vector acting as the acceptance of past information.
- W: **Weight** is the (trainable) positional weight decay vector.
- K: **Key**, analogous to K in standard attention.
- V: **Value**, analogous to V in standard attention.



RWKV: architecture overview

- R: **Receptance** vector acting as the acceptance of past information.
- W: **Weight** is the (trainable) positional weight decay vector.
- K: **Key**, analogous to K in standard attention.
- V: **Value**, analogous to V in standard attention.

Sigmoid
(forget gate)

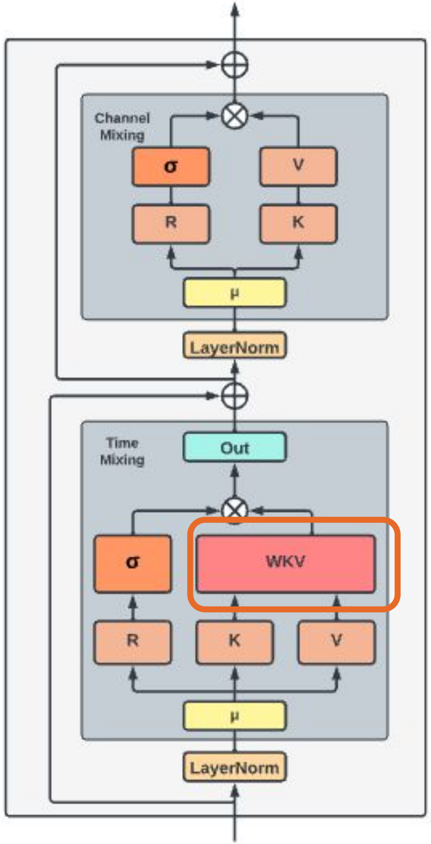


RWKV: architecture overview

- R: **Receptance** vector acting as the acceptance of past information.
- W: **Weight** is the (trainable) positional weight decay vector.
- K: **Key**, analogous to K in standard attention.
- V: **Value**, analogous to V in standard attention.

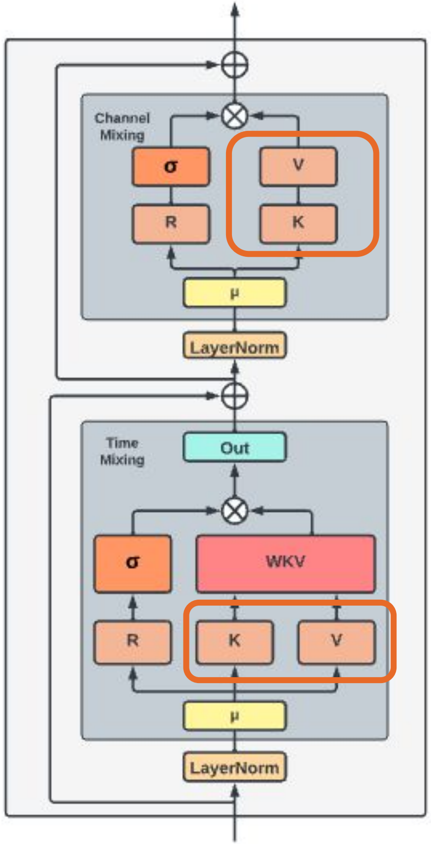
$$w_{t,i} = -(t-i)w, \tag{10}$$

where $w \in (R_{\geq 0})^d$, with d the number of channels. We require w to be non-negative to ensure that $e^{w_{t,i}} \leq 1$ and the per-channel weights decay backwards in time.



RWKV: architecture overview

- R: **Receptance** vector acting as the acceptance of past information.
- W: **Weight** is the (trainable) positional weight decay vector.
- K: **Key**, analogous to K in standard attention.
- V: **Value**, analogous to V in standard attention.



Time-Mixing Block

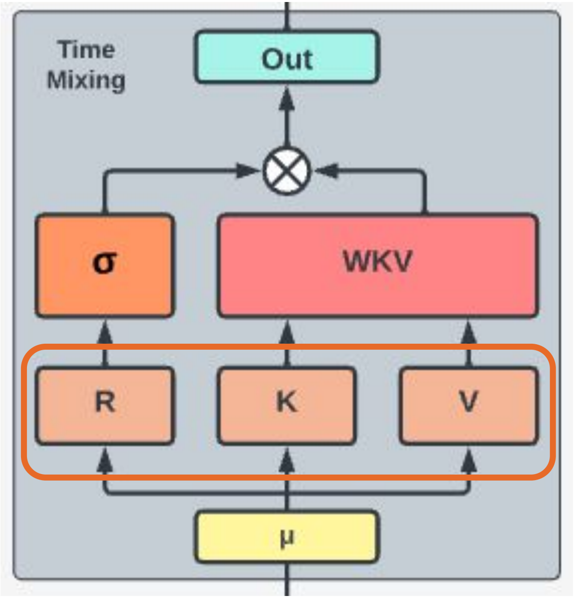
$$r_t = W_r \cdot (\mu_r x_t + (1 - \mu_r) x_{t-1}), \tag{11}$$

$$k_t = W_k \cdot (\mu_k x_t + (1 - \mu_k) x_{t-1}), \tag{12}$$

$$v_t = W_v \cdot (\mu_v x_t + (1 - \mu_v) x_{t-1}), \tag{13}$$

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}}, \tag{14}$$

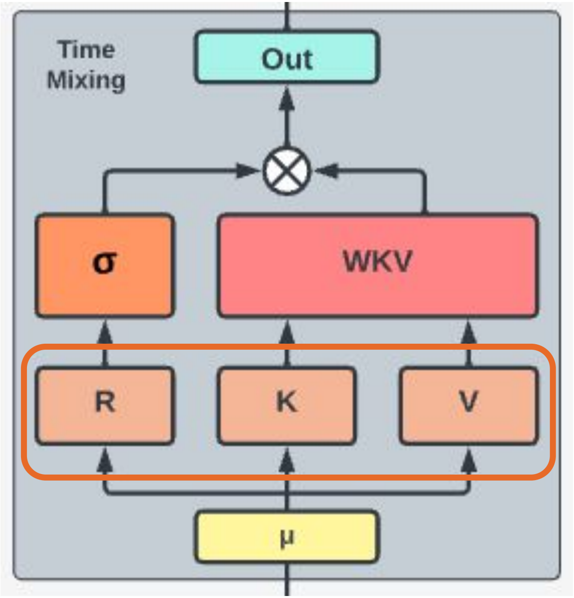
$$o_t = W_o \cdot (\sigma(r_t) \odot wkv_t), \tag{15}$$



Time-Mixing Block

Analogous to
standard transformer

$$
\begin{aligned}
r_t &= \boxed{W_r \cdot (\mu_r x_t} + (1-\mu_r)x_{t-1}), && (11)\\
k_t &= \boxed{W_k \cdot (\mu_k x_t} + (1-\mu_k)x_{t-1}), && (12)\\
v_t &= \boxed{W_v \cdot (\mu_v x_t} + (1-\mu_v)x_{t-1}), && (13)\\
wkv_t &= \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}}, && (14)\\
o_t &= W_o \cdot (\sigma(r_t) \odot wkv_t), && (15)
\end{aligned}
$$



Time-Mixing Block

Token Shift

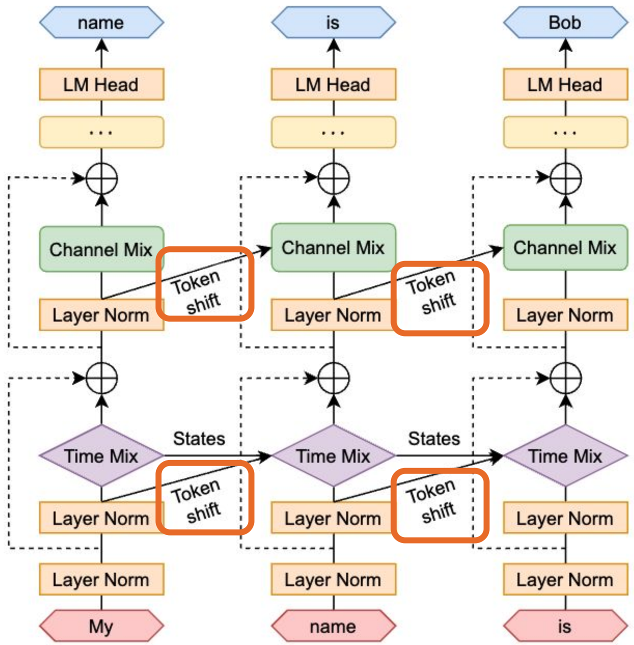
$$r_t = W_r \cdot (\mu_r x_t + \boxed{(1-\mu_r)x_{t-1}}), \quad (11)$$

$$k_t = W_k \cdot (\mu_k x_t + \boxed{(1-\mu_k)x_{t-1}}), \quad (12)$$

$$v_t = W_v \cdot (\mu_v x_t + \boxed{(1-\mu_v)x_{t-1}}), \quad (13)$$

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}}, \quad (14)$$

$$o_t = W_o \cdot (\sigma(r_t) \odot wkv_t), \quad (15)$$



Time-Mixing Block

$$r_t = W_r \cdot (\mu_r x_t + (1-\mu_r)x_{t-1}), \tag{11}$$

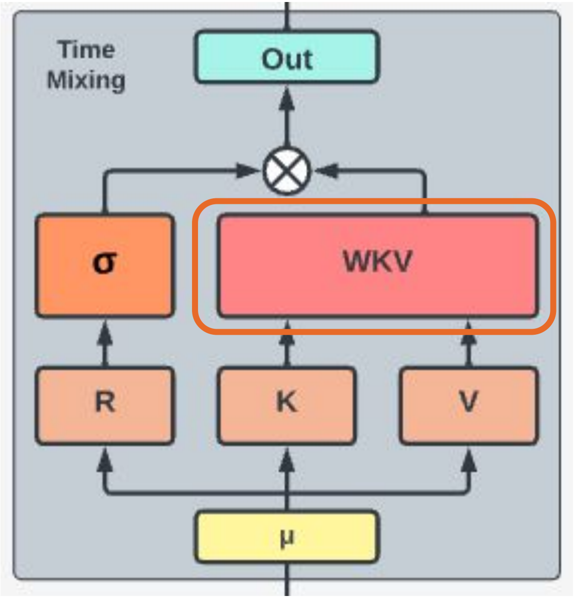
$$k_t = W_k \cdot (\mu_k x_t + (1-\mu_k)x_{t-1}), \tag{12}$$

$$v_t = W_v \cdot (\mu_v x_t + (1-\mu_v)x_{t-1}), \tag{13}$$

$$wkv_t = \boxed{\frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}}}, \tag{14}$$

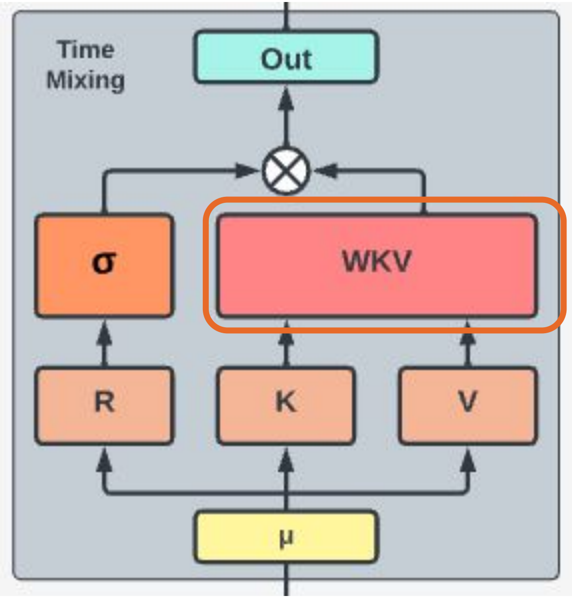
$$o_t = W_o \cdot (\sigma(r_t) \odot wkv_t), \tag{15}$$

"Attention" in RWKV



WKV Computation

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}}, \quad (14)$$

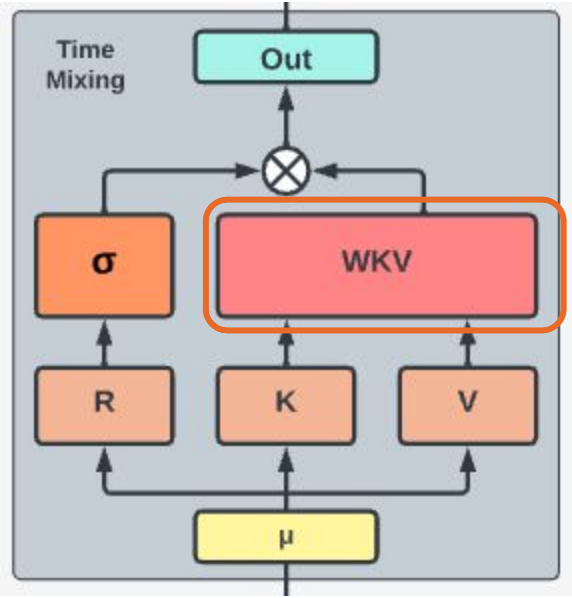


WKV Computation

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{\boxed{-(t-1-i)w}+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{\boxed{-(t-1-i)w}+k_i} + e^{u+k_t}}, \quad (14)$$

$$w_{t,i} = -(t-i)w, \quad (10)$$

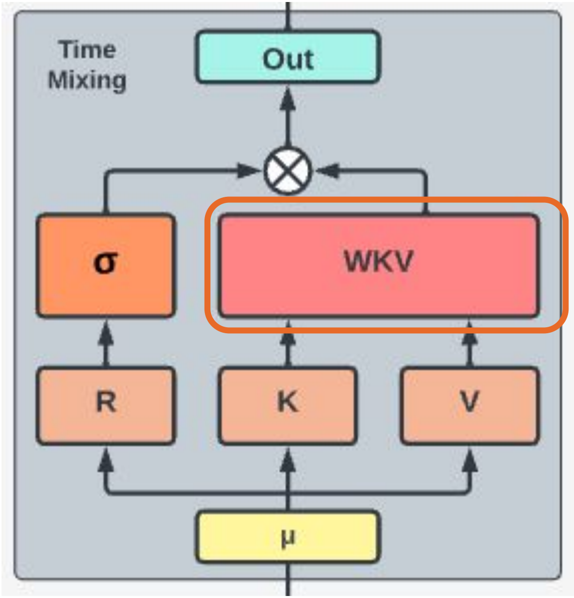
where $w \in (R_{\geq 0})^d$, with d the number of channels. We require w to be non-negative to ensure that $e^{w_{t,i}} \leq 1$ and the per-channel weights decay backwards in time.



WKV Computation

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + \boxed{e^{u+k_t}} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}}, \quad (14)$$

Engineering trick: special treatment for the current timestep t. Need an additional learned u vector to “compensate for degradation of w”.



Time-Mixing Block

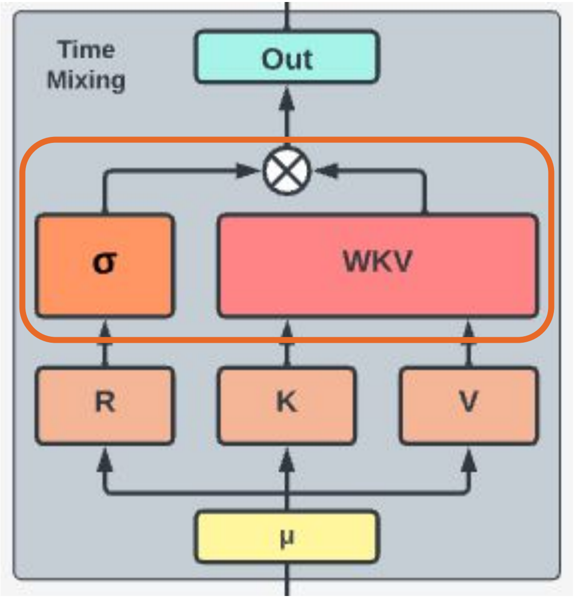
$$r_t = W_r \cdot (\mu_r x_t + (1 - \mu_r) x_{t-1}), \tag{11}$$

$$k_t = W_k \cdot (\mu_k x_t + (1 - \mu_k) x_{t-1}), \tag{12}$$

$$v_t = W_v \cdot (\mu_v x_t + (1 - \mu_v) x_{t-1}), \tag{13}$$

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}}, \tag{14}$$

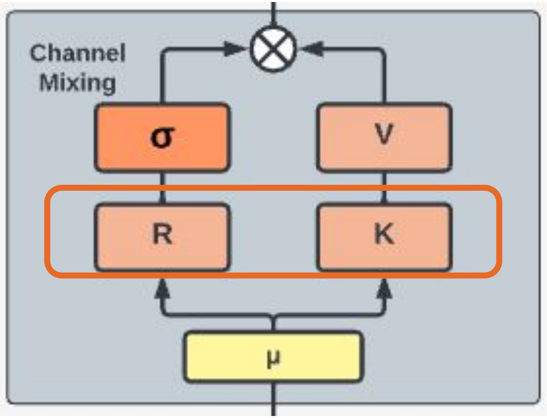
$$o_t = \boxed{W_o \cdot (\sigma(r_t) \odot wkv_t),} \tag{15}$$



Channel-Mixing Block

$$\boxed{\begin{aligned} r_t &= W_r \cdot (\mu_r x_t + (1 - \mu_r) x_{t-1}), \\ k_t &= W_k \cdot (\mu_k x_t + (1 - \mu_k) x_{t-1}), \end{aligned}}$$

$$o_t = \sigma(r_t) \odot (W_v \cdot \max(k_t, 0)^2),$$

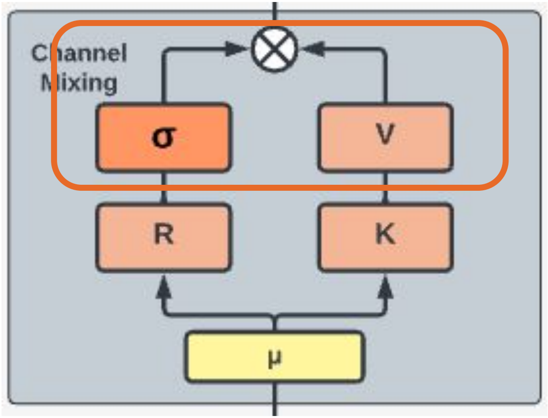


Channel-Mixing Block

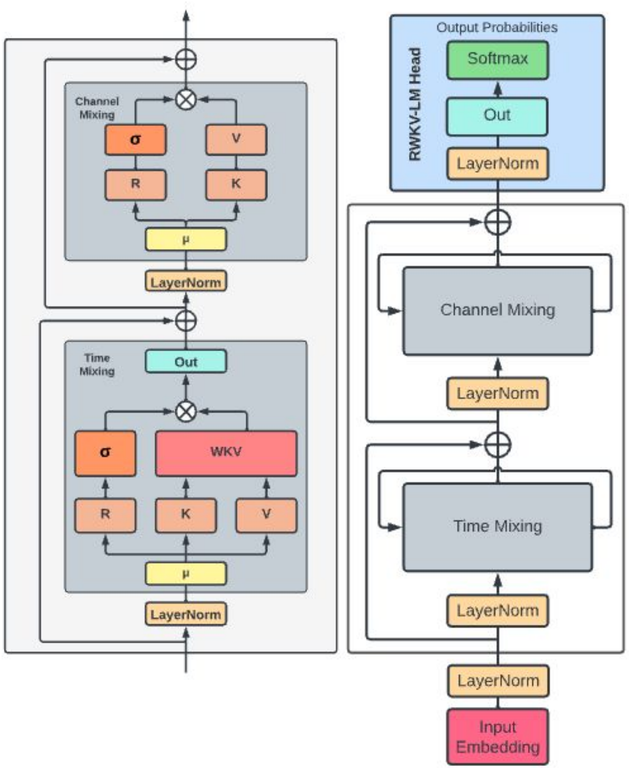
$$r_t = W_r \cdot (\mu_r x_t + (1 - \mu_r) x_{t-1}),$$

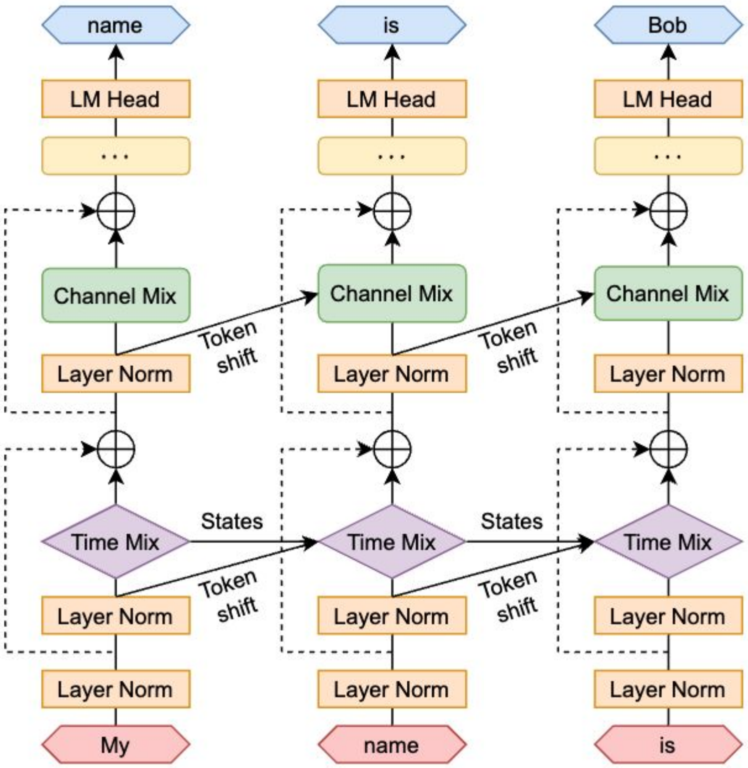
$$k_t = W_k \cdot (\mu_k x_t + (1 - \mu_k) x_{t-1}),$$

$$\boxed{o_t = \sigma(r_t) \odot (W_v \cdot \max(k_t, 0)^2),}$$



Putting them together:





Notes on efficiency

$$\mathrm{Attn}(Q,K,V)_t = \frac{\sum_{i=1}^{T} e^{\boxed{q_t^\top k_i}} v_i}{\sum_{i=1}^{T} e^{q_t^\top k_i}}. \qquad (8)$$

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}}, \quad (14)$$

- Given T tokens, d hidden dimension.
- Standard attention: O(T^2d).

Notes on efficiency

$$\mathrm{Attn}(Q,K,V)_t = \frac{\sum_{i=1}^{T} e^{q_t^\top k_i} v_i}{\sum_{i=1}^{T} e^{q_t^\top k_i}}. \qquad (8)$$

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} \boxed{v_i} + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}}, \quad (14)$$

- Given T tokens, d hidden dimension.
- Standard attention: O(T^2d).
- WKV:
    - We can reuse v and k!

Notes on efficiency

$$\mathrm{Attn}(Q,K,V)_t = \frac{\sum_{i=1}^{T} e^{q_t^\top k_i} v_i}{\sum_{i=1}^{T} e^{q_t^\top k_i}}. \tag{8}$$

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + \boxed{e^{u+k_t} v_t}}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + \boxed{e^{u+k_t}}}, \tag{14}$$

- Given T tokens, d hidden dimension.
- Standard attention: O(T^2d).
- WKV:
  - We can reuse v and k!
  - Compute (update) the wkv score costs O(Td).

Express WKV recursively

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}}, \quad (14)$$

$$\downarrow$$

$$wkv_t = \frac{a_{t-1} + e^{u+k_t} v_t}{b_{t-1} + e^{u+k_t}}, \qquad (20)$$

$$a_t = e^{-w} a_{t-1} + e^{k_t} v_t, \qquad (21)$$

$$b_t = e^{-w} b_{t-1} + e^{k_t}. \qquad (22)$$

Express WKV recursively

$$wkv_t = \frac{a_{t-1} + e^{u+k_t} v_t}{b_{t-1} + e^{u+k_t}}, \tag{20}$$

$$a_t = e^{-w} a_{t-1} + e^{k_t} v_t, \tag{21}$$

$$b_t = e^{-w} b_{t-1} + e^{k_t}. \tag{22}$$

Key idea: WKV can be viewed both as attention and RNN-like hidden state.

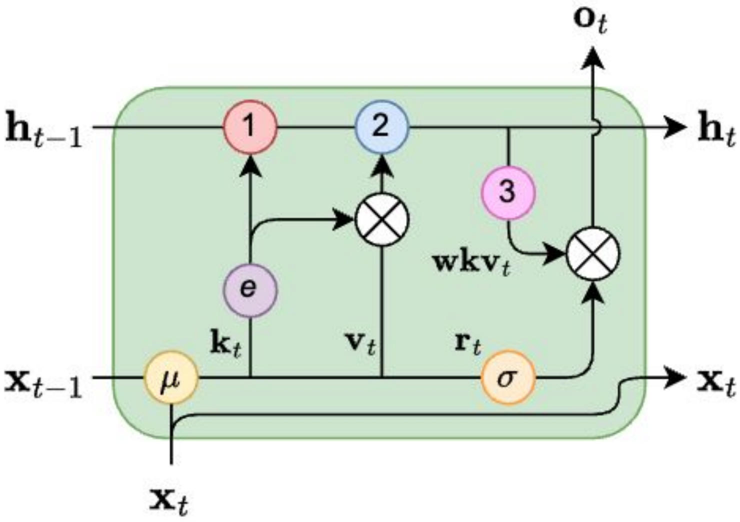
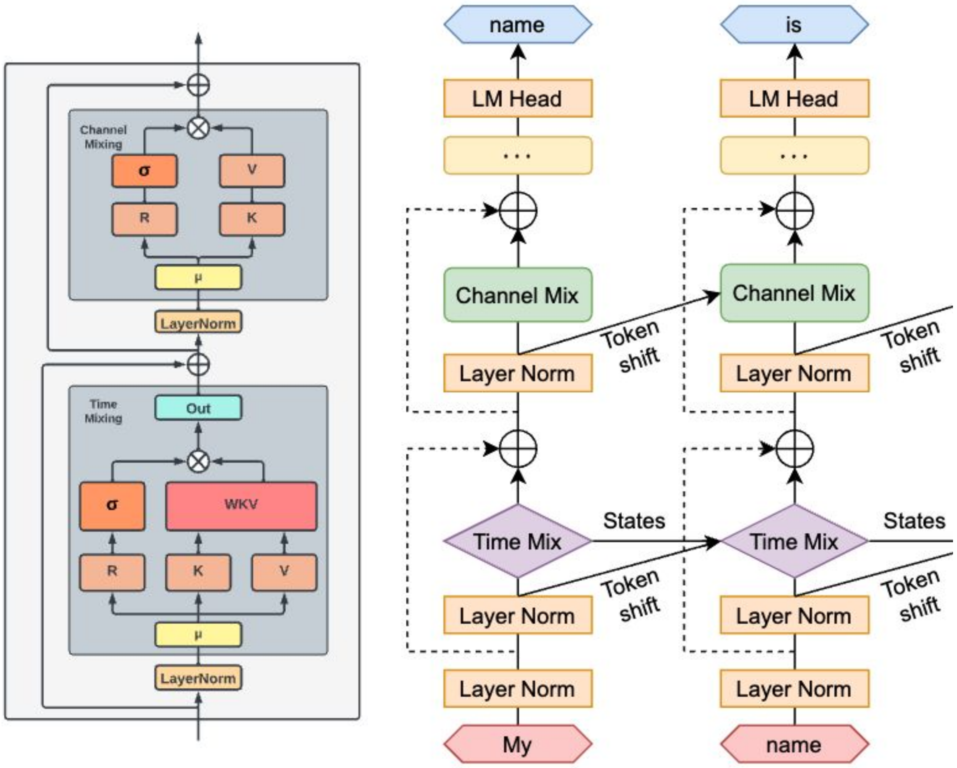


Figure 7: RWKV time-mixing block formulated as an RNN cell. Color codes: yellow (μ) denotes the token shift, red (1) denotes the denominator, blue (2) denotes the numerator, pink (3) denotes the fraction computations in 14. h denotes the numerator-denominator tuple (a, b).

Training: time-parallel mode

Similar to Transformer: take advantage of parallelized training.

- Wr, Wk, Wv matrices are easily parallelizable.
- WKV is time-dependent, but can be parallelized along axis of batch and dimension.
- Further, token shift is implemented easily with `nn.ZeroPad2d((0,0,1,-1))`

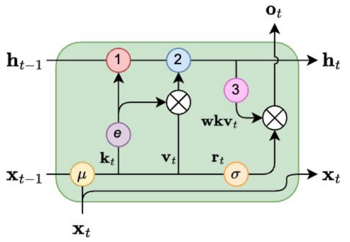


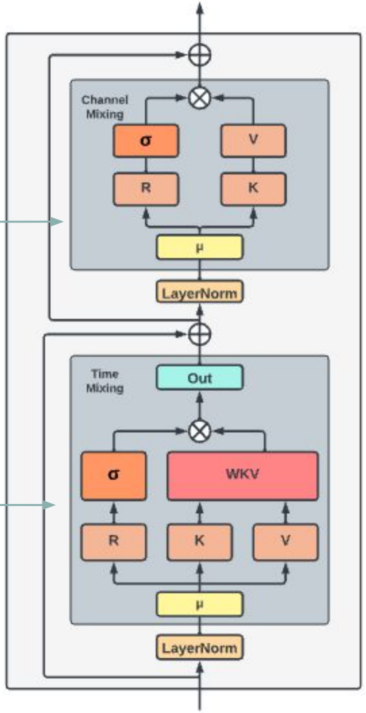
Inference: time-sequence mode

Similar to RNN: take advantage of constant memory footprint.

- Replace the time-mixing block with the equivalent RNN cell.
- Constant speed and memory footprint, regardless of sequence length.
- In contrast, self-attention requires linearly growing KV cache for increasing sequence length.

$$r_t = W_r \cdot (\mu_r x_t + (1 - \mu_r) x_{t-1}),$$
$$k_t = W_k \cdot (\mu_k x_t + (1 - \mu_k) x_{t-1}),$$
$$o_t = \sigma(r_t) \odot (W_v \cdot \max(k_t, 0)^2),$$





Experiments

- **RQ1:** Is RWKV competitive against quadratic transformer architectures with equal number of parameters and training tokens?
- **RQ2:** When increasing the number of parameters, does RWKV remain competitive against quadratic transformer architectures?
- **RQ3:** Does increasing parameters of RWKV yield better language modeling loss, when RWKV models are trained for context lengths that most open-sourced quadratic transformers *cannot* efficiently process?

Performance and scaling behavior

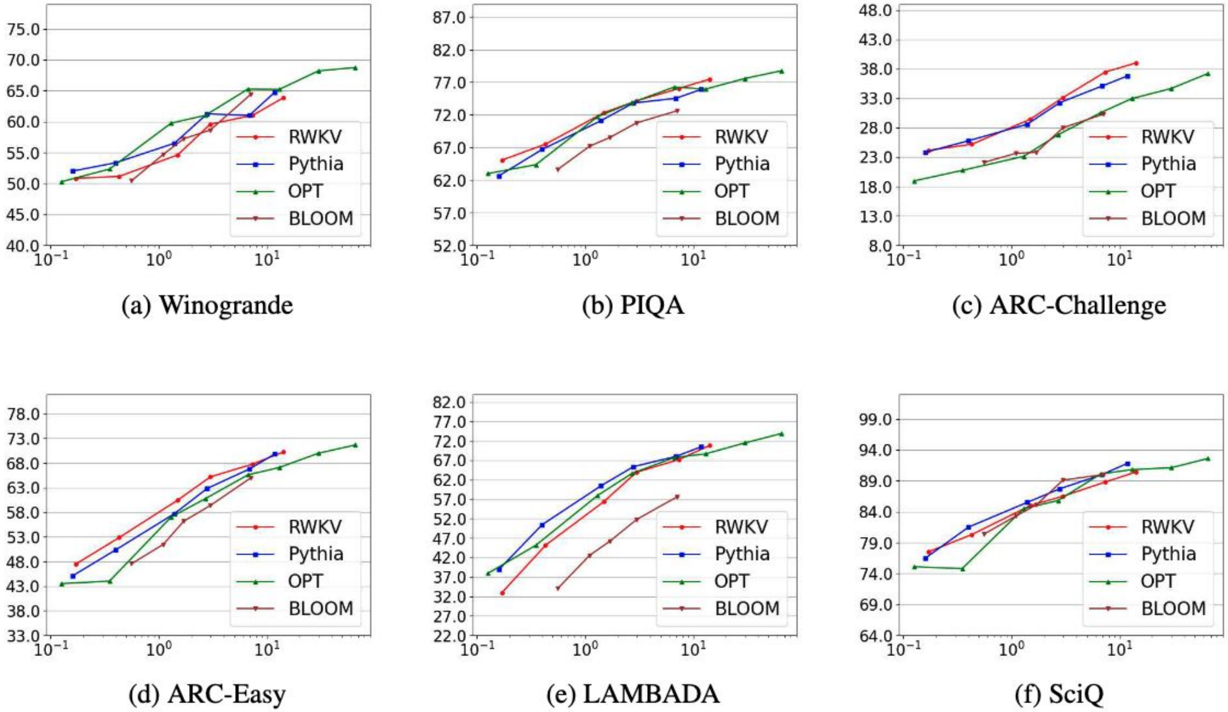


Figure 4: Zero-Shot Performance: The horizontal axis is a number of parameters and the vertical axis is accuracy.

Effect of context length

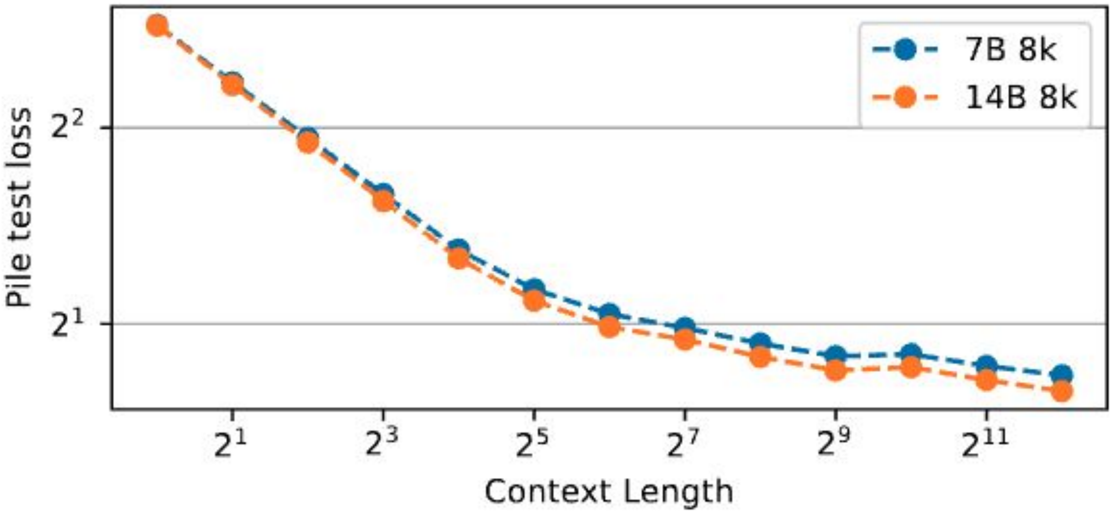


Figure 5: Increasing context length contributes to lower test loss on the Pile (Gao et al., 2020).

Inference efficiency

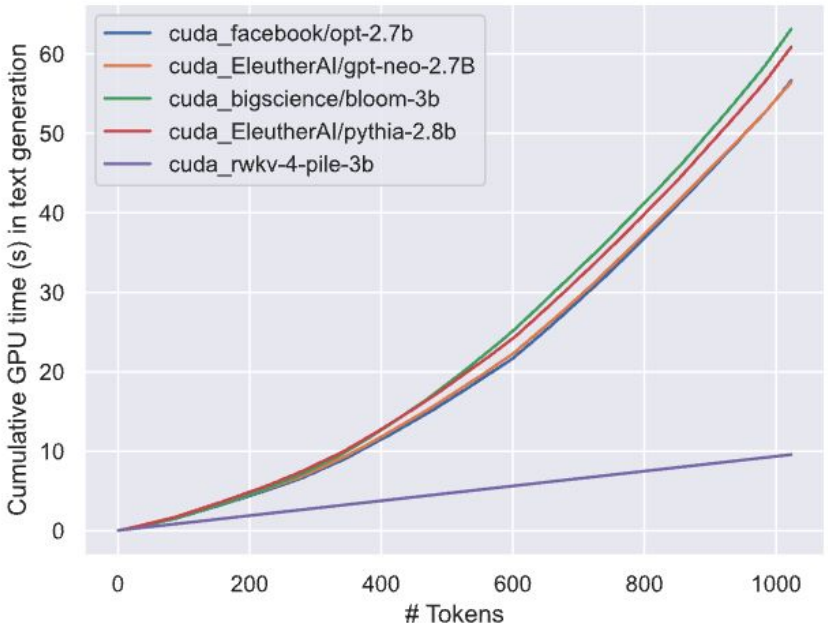


Figure 6: Cumulative time during text generation for different LLMs.

Observation on prompts

Additionally, we carried out comparative studies on RWKV-4 and ChatGPT / GPT-4, see Appendix J. They revealed that RWKV-4 is very sensitive to prompt engineering. When the prompts were adjusted from the ones used for GPT to more suitable for RWKV, the F1-measure performance increased even from 44.2% to 74.8%.

Prompt for ChatGPT:

Having premise <here is a premise> judge if the following hypothesis <here is a hypothesis> are logically connected with the premise? Answer "entailment" if yes, or "not_entailment" if no.

Prompt for RWKV:

Can you tell me if the hypothesis is entailment or is not entailment to the premise?

premise: <here is a premise>
hypothesis: <here is a hypothesis>

Takeaway

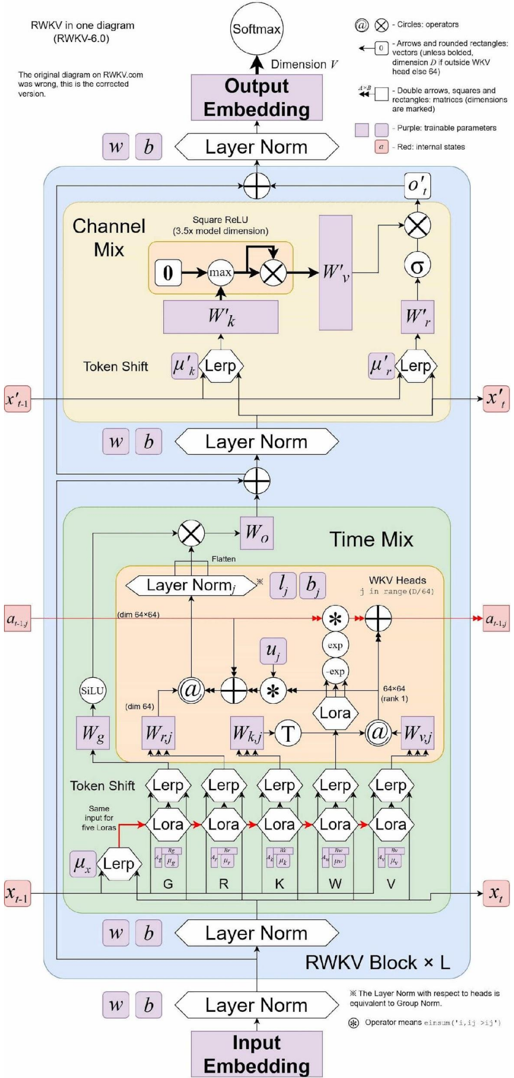
Main Contributions:

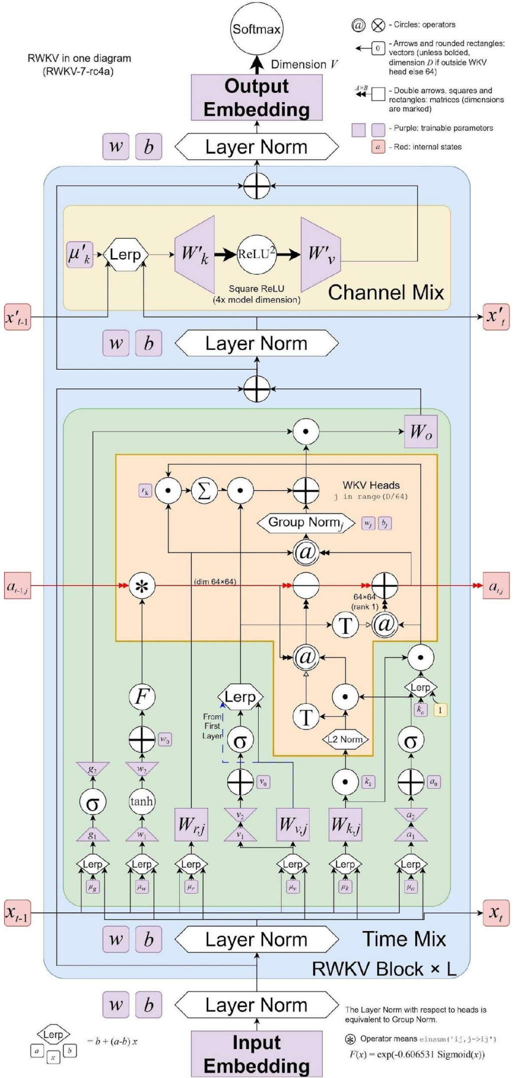
- A linear "attention" mechanism that replaces the quadratic self-attention.
- Enjoy both parallelization (for training) and constant speed/memory footprint (for inference).

Limitations:

- RNN-style single vector representation limits the model's ability to "look back", compared to exact information retained in self-attention.
- Consequently, prompt engineering is crucial to the model performance.

Rapidly Evolving Architecture of RWKV





Benchmark Performance of RWKV-7 (circa Feb 2025)

*** RWKV-7 World-v3 1.5B MMLU 44.84% (organic leap from RWKV-6 World-v2.1 1.6B MMLU 26.34%, no eval-maxxing, no HQ-annealing, no post-training) ***

lm-evaluation-harness	params	LAMBADA	English	LAMBADA	PIQA	StoryCloze16	Hellaswag	WinoGrande	arc_challeng	arc_easy	headQA_en	openbookQA	sciq	ReCoRD	MultiLang	xLBD	xSC	xWG	xCOPA
model	B	ppl	avg%	acc	acc	acc	acc_norm	acc	acc_norm	acc	acc_norm	acc_norm	acc	em	avg%	acc	acc	acc	acc
RWKV-7 "Goose" World-v3	1.52	4.13	67.7%	69.4%	77.2%	76.3%	70.8%	67.9%	42.5%	75.8%	39.0%	46.4%	94.4%	85.3%	58.6%	43.0%	59.7%	72.2%	59.7%
SmolLM2-1.7B (+emb = 1.81b)	1.81	4.45	67.9%	67.4%	76.7%	76.0%	71.5%	66.7%	47.0%	78.0%	39.8%	44.6%	93.4%	86.3%	49.2%	29.3%	52.0%	62.5%	53.1%
Qwen2.5-1.5B (+emb = 1.78b)	1.78	5.68	65.5%	62.2%	75.8%	73.8%	67.9%	63.5%	45.3%	75.4%	37.6%	40.2%	94.6%	83.9%	53.8%	34.1%	55.9%	68.1%	57.1%
stablelm-2-1_6b	1.64	4.94	65.1%	65.8%	76.3%	75.8%	68.9%	64.3%	39.9%	68.6%	34.3%	39.8%	95.3%	86.8%	52.3%	34.8%	53.8%	64.4%	56.3%
RWKV-6 "Finch" World-v2.1	1.60	4.58	62.2%	67.3%	73.9%	74.9%	61.1%	60.5%	34.2%	64.2%	35.7%	38.6%	90.3%	83.7%	56.2%	40.9%	56.7%	69.2%	58.1%

lm-evaluation-harness	params	LAMBADA	English	LAMBADA	PIQA	StoryCloze16	Hellaswag	WinoGrande	arc_challeng	arc_easy	headQA_en	openbookQA	sciq	ReCoRD	MultiLang	xLBD	xSC	xWG	xCOPA
model	B	ppl	avg%	acc	acc	acc	acc_norm	acc	acc_norm	acc	acc_norm	acc_norm	acc	em	avg%	acc	acc	acc	acc
RWKV-7 "Goose" World-v2.9	0.450	6.94	59.9%	58.5%	72.9%	70.5%	56.8%	60.3%	33.0%	66.0%	33.5%	38.2%	89.8%	79.3%	52.4%	34.8%	54.1%	65.3%	55.3%
SmolLM2-360M (+emb = 407m)	0.407	9.36	59.7%	53.4%	71.8%	68.2%	56.4%	59.0%	37.9%	70.2%	34.3%	37.8%	90.8%	77.4%	43.7%	18.7%	49.7%	54.5%	51.8%
RWKV-7 "Goose" Pile	0.421	7.21	55.8%	57.9%	69.2%	67.7%	48.1%	56.4%	27.6%	56.2%	32.1%	32.2%	85.9%	80.2%	47.6%	29.4%	50.7%	57.3%	52.9%
mamba-370m (+emb = 421m) Pile	0.421	8.14	54.7%	55.6%	69.5%	66.3%	46.5%	55.5%	27.9%	55.0%	32.3%	30.8%	84.9%	77.0%	47.2%	28.5%	50.5%	57.3%	52.4%
RWKV-5 "Eagle" World-v2	0.462	8.87	53.1%	54.0%	66.5%	65.7%	40.9%	53.1%	26.3%	54.0%	30.9%	31.2%	86.6%	75.0%	49.5%	32.5%	52.8%	58.8%	54.0%

lm-evaluation-harness	params	LAMBADA	English	LAMBADA	PIQA	StoryCloze16	Hellaswag	WinoGrande	arc_challeng	arc_easy	headQA_en	openbookQA	sciq	ReCoRD	MultiLang	xLBD	xSC	xWG	xCOPA
model	B	ppl	avg%	acc	acc	acc	acc_norm	acc	acc_norm	acc	acc_norm	acc_norm	acc	em	avg%	acc	acc	acc	acc
RWKV-7 "Goose" World-v2.8	0.191	12.4	52.6%	49.1%	67.1%	63.7%	42.2%	52.5%	27.6%	56.7%	29.9%	33.0%	85.5%	71.3%	47.5%	27.3%	51.5%	58.1%	53.0%
SmolLM2-135M (+emb = 163m)	0.163	19.0	53.0%	42.9%	68.2%	63.4%	43.2%	53.1%	29.7%	64.3%	31.1%	33.0%	83.8%	70.1%	41.7%	12.5%	49.4%	52.8%	52.0%
RWKV-7 "Goose" Pile	0.168	14.2	49.8%	45.6%	65.5%	61.8%	36.9%	52.3%	24.7%	47.9%	29.1%	30.0%	81.8%	71.9%	43.9%	21.7%	49.3%	52.0%	52.8%
mamba-130m (+emb = 168m) Pile	0.168	16.0	48.5%	44.2%	64.4%	60.4%	35.3%	52.4%	24.3%	48.1%	28.8%	28.6%	78.1%	68.9%	43.7%	20.1%	49.2%	53.2%	52.4%

More Benchmark Performance of RWKV-7 (circa Feb 2025)

lm-evaluation-harness	params	LAMBADA	English	LAMBADA	PIQA	StoryCloze16	Hellaswag	WinoGrande	arc_challeng	arc_easy	headQA_en	openbookQA	sciq	ReCoRD
(same dataset & tokenizer)	B	ppl	avg%	acc	acc	acc	acc_norm	acc	acc_norm	acc	acc_norm	acc_norm	acc	em
RWKV-7-Pile "Goose"	**1.47**	4.80	62.5%	67.0%	73.5%	73.0%	61.8%	64.6%	33.3%	64.8%	36.1%	35.8%	91.5%	85.8%
mamba-1.4b (1.47b if +emb)	1.47	5.04	61.0%	64.9%	74.2%	70.8%	59.1%	61.3%	33.0%	65.5%	36.0%	36.4%	87.1%	83.2%
pythia-1.4b-v0	1.41	6.58	56.8%	60.4%	71.2%	67.7%	50.8%	57.0%	28.6%	57.7%	33.3%	31.0%	85.5%	81.4%
RWKV-4-Pile "Dove"	1.52	6.91	56.8%	57.4%	72.1%	68.0%	52.8%	54.7%	29.2%	60.6%	34.8%	34.0%	84.2%	77.0%

lm-evaluation-harness	params	LAMBADA	English	LAMBADA	PIQA	StoryCloze16	Hellaswag	WinoGrande	arc_challeng	arc_easy	headQA_en	openbookQA	sciq	ReCoRD
(same dataset & tokenizer)	B	ppl	avg%	acc	acc	acc	acc_norm	acc	acc_norm	acc	acc_norm	acc_norm	acc	em
RWKV-7-Pile "Goose"	**0.421**	7.21	55.8%	57.9%	69.2%	67.7%	48.1%	56.4%	27.6%	56.2%	32.1%	32.2%	85.9%	80.2%
mamba-370m (421m if +emb)	0.421	8.14	54.7%	55.6%	69.5%	66.3%	46.5%	55.5%	27.9%	55.0%	32.3%	30.8%	84.9%	77.0%
pythia-410m-v0	0.405	11.7	51.3%	50.2%	66.8%	62.5%	39.1%	53.0%	26.0%	50.6%	31.2%	29.6%	81.1%	74.8%
RWKV-4-Pile "Dove"	0.430	13.1	51.1%	45.0%	67.7%	63.8%	40.9%	51.9%	25.3%	52.7%	31.4%	32.6%	80.3%	70.3%

lm-evaluation-harness	params	LAMBADA	English	LAMBADA	PIQA	StoryCloze16	Hellaswag	WinoGrande	arc_challeng	arc_easy	headQA_en	openbookQA	sciq	ReCoRD
(same dataset & tokenizer)	B	ppl	avg%	acc	acc	acc	acc_norm	acc	acc_norm	acc	acc_norm	acc_norm	acc	em
RWKV-7-Pile L33-D512	0.164	12.7	50.7%	49.0%	65.7%	62.6%	38.3%	51.3%	25.3%	49.6%	29.1%	30.4%	83.0%	73.6%
RWKV-7-Pile L25-D576	0.165	12.2	50.6%	49.5%	66.5%	62.3%	38.2%	52.5%	24.0%	49.6%	28.4%	29.6%	82.9%	73.0%
RWKV-7-Pile "Goose"	**0.168**	14.2	49.8%	45.6%	65.5%	61.8%	36.9%	52.3%	24.7%	47.9%	29.1%	30.0%	81.8%	71.9%
mamba-130m (168m if +emb)	0.168	16.0	48.5%	44.2%	64.4%	60.4%	35.3%	52.4%	24.3%	48.1%	28.8%	28.6%	78.1%	68.9%
pythia-160m-v0	0.162	24.4	46.7%	38.8%	62.6%	58.4%	31.7%	52.0%	24.0%	45.3%	28.7%	29.0%	76.5%	66.3%
RWKV-4-Pile "Dove"	0.169	29.2	46.2%	33.1%	64.9%	59.1%	32.2%	51.5%	23.9%	47.1%	28.3%	29.4%	77.2%	61.9%

References on RWKV

- James Bradbury et al (from Salesforce Research), “Quasi-Recurrent Neural Networks,” arxiv:1611.10576v2, Nov 2016.
- Shuangfei Zhai et al (from Apple Computer), ”An Attention Free Transformer” (AFT), arxiv:2105.14103, May 2021.
- Bo Peng et al, “RWKV: Reinventing RNNs for the Transformer Era,” arxiv:2305:13048, Findings of EMNLP 2023.
- Bo Peng et al, “Eagle and Finch: RWKV withMatrix-Valued States and Dynamic Recurrence”, arXiv:2404.05892v4, Sept 2024, [Interesting discussions in the “Limitations” section of this paper.]
- Zhiyuan Li, Tingyu Xia, Yi Chang and Yuan Wu, “A Survey of RWKV”, arxiv2412.14847v2, Jan 2025
- https://huggingface.co/blog/rwkv
- https://www.rwkv.com
- Short Talk by Eugene Cheah - From idea to LLM (RWKV / Recursal) https://www.youtube.com/watch?v=yiewqC6qNM8
- Reading the Original RWKV paper with Yannic Kilcher RWKV: Reinventing RNNs for the Transformer Era (Paper Explained), https://www.youtube.com/watch?v=x8pW19wKfXQ
- 侯皓文：RWKV论文解读 - 在Transformer时代重塑RNN, https://www.youtube.com/watch?v=oIuAv99GHBs
- Deep Dive on RWKV: 2-hour interview w/ Eugene Cheah, an RWKV committee member: https://youtu.be/dvk6X5zeIfY

2024 in Post-Transformers Architectures (State Space Models, RWKV) [LS Live @ NeurIPS]

https://www.latent.space/p/2024-post-transformers