模式分析与机器智能
工业和信息化部重点实验室
MIIT Key Laboratory of
Pattern Analysis & Machine Intelligence

ParNeC 模式识别与神经计算研究组
PAttern Recognition and NEural Computing

In-depth Introduction to the Mamba

欣子豪

2024.6.24

Contents

1. FlashAttention
GPU Memory Hierarchy

2. State Space Mode

3. HiPPO

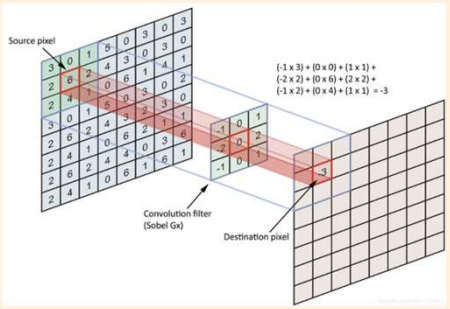
4. Structured State Space Models (S4)

5. Mamba

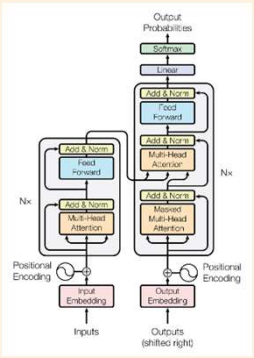
6. Mamba for Computer Vision

Approach to Efficiency

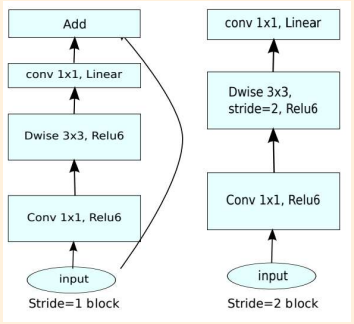
Fundamental algorithms



CNN



Transformer



MobileNet

Hardware

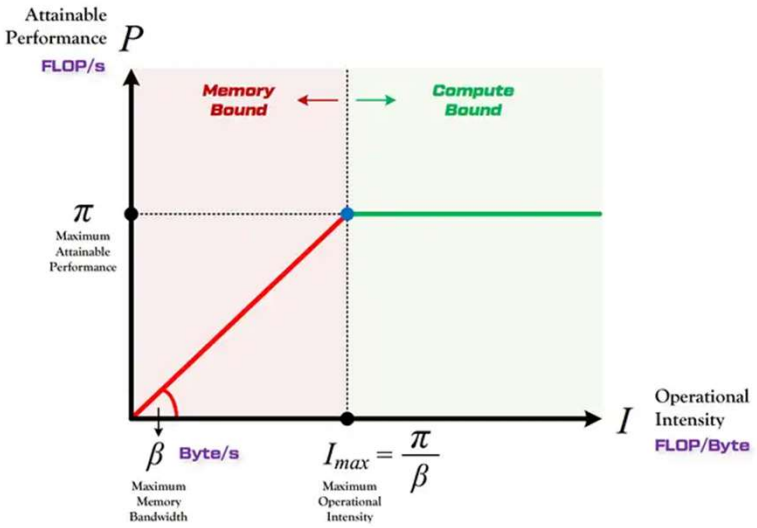
GPU

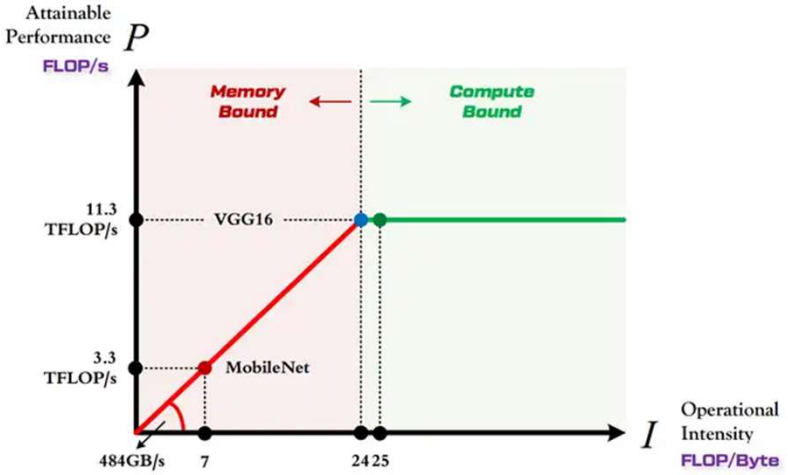
CPU

Edge device

Approach to Efficiency

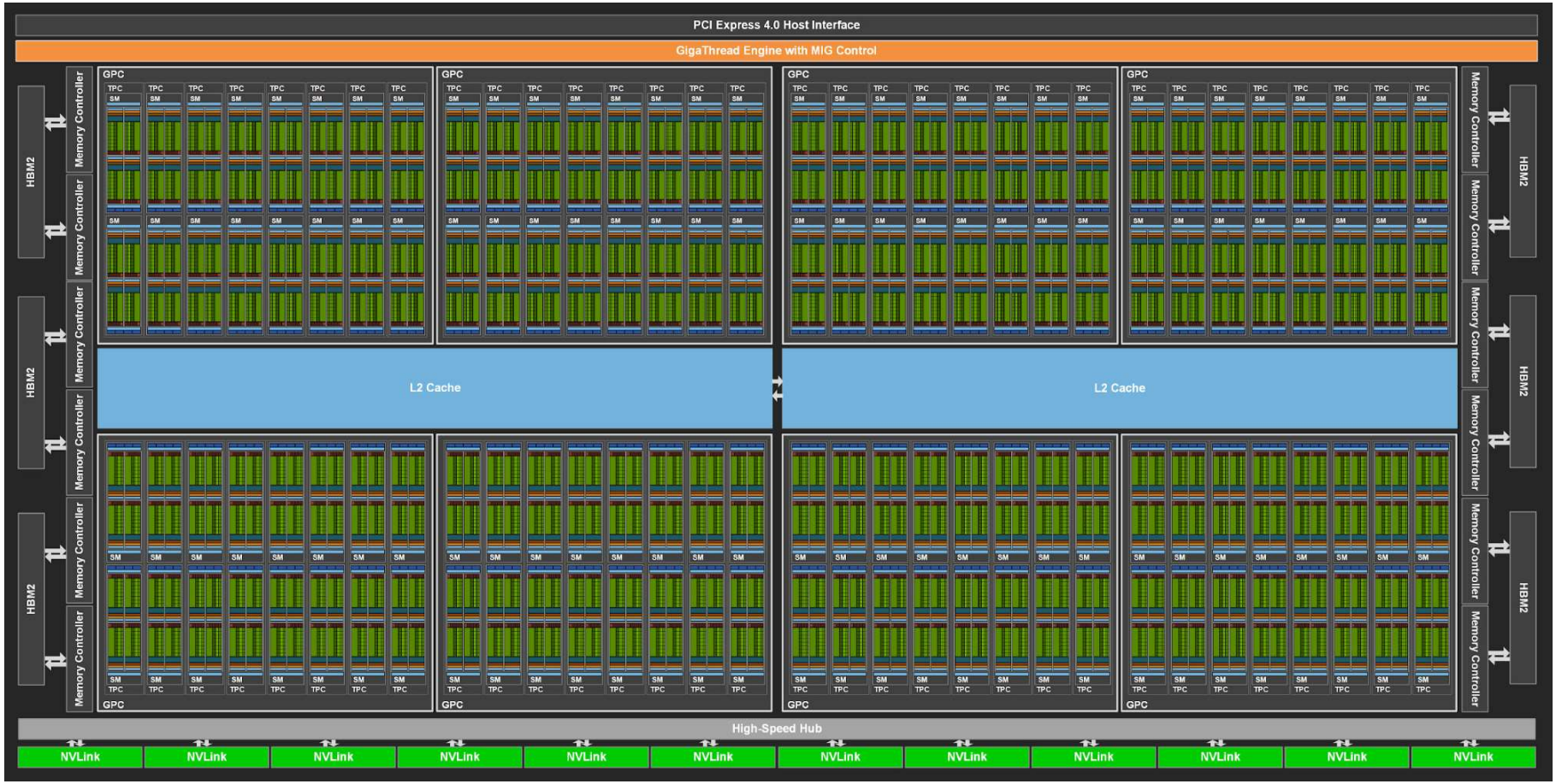
Roofline Model - Operational Intensity





https://www.cnblogs.com/studyinglover/p/18127962

How the GPU Calculates

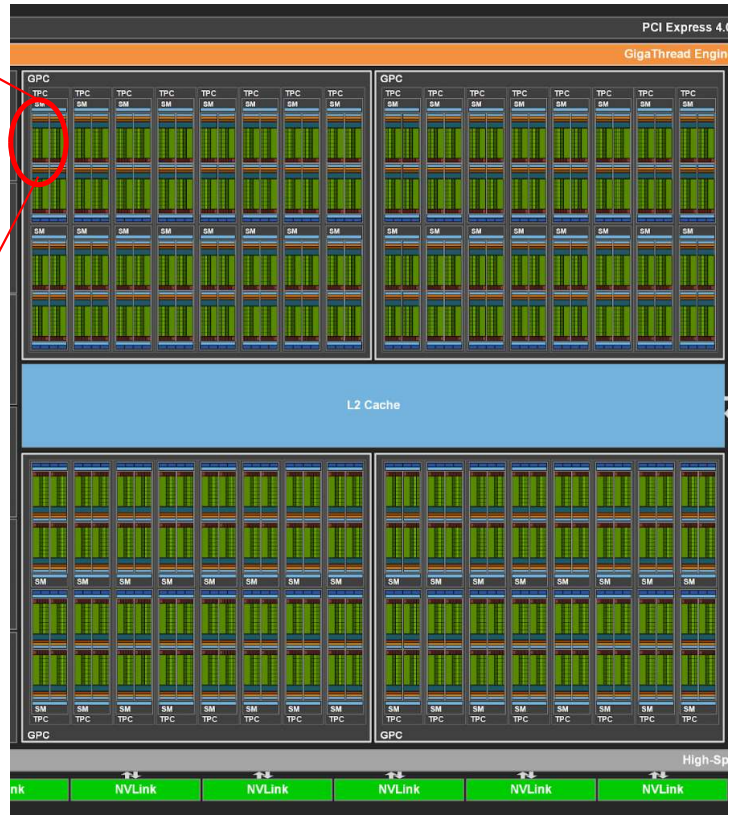


A100 GPU with 128 SMs

https://resources.nvidia.com/en-us-genomics-ep/ampere-architecture-white-paper?xs=169656

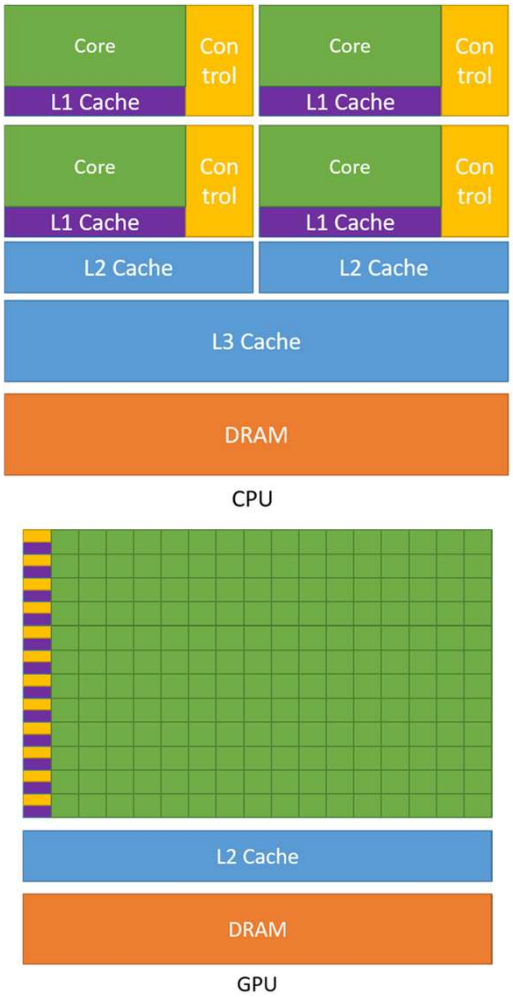
How the GPU Calculates

A100 GPU Streaming Multiprocessor(SM)

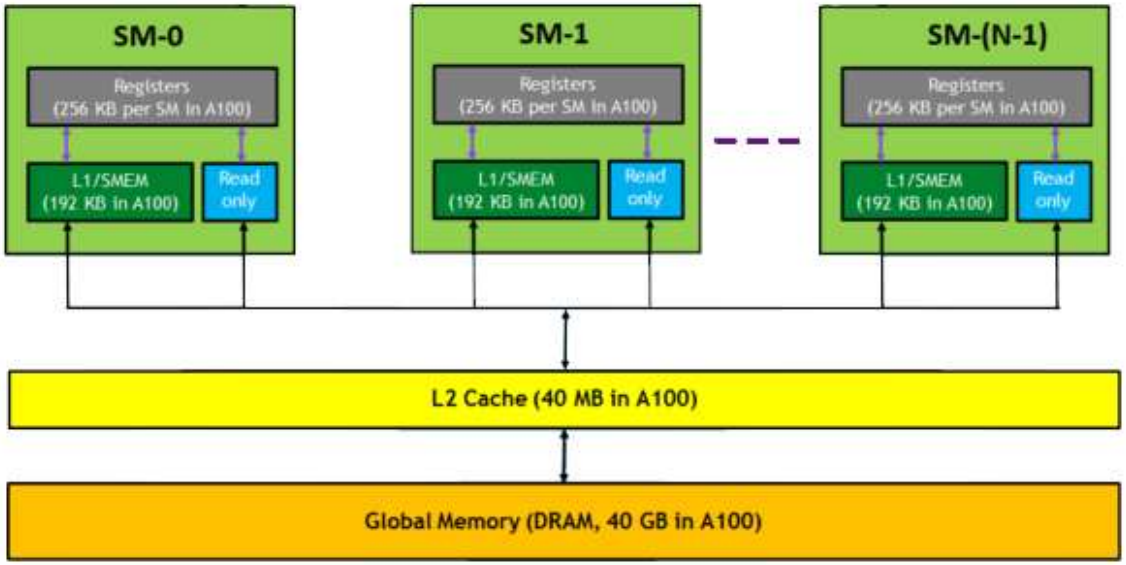


https://resources.nvidia.com/en-us-genomics-ep/ampere-architecture-white-paper?xs=169656

GPU Memory Hierarchy

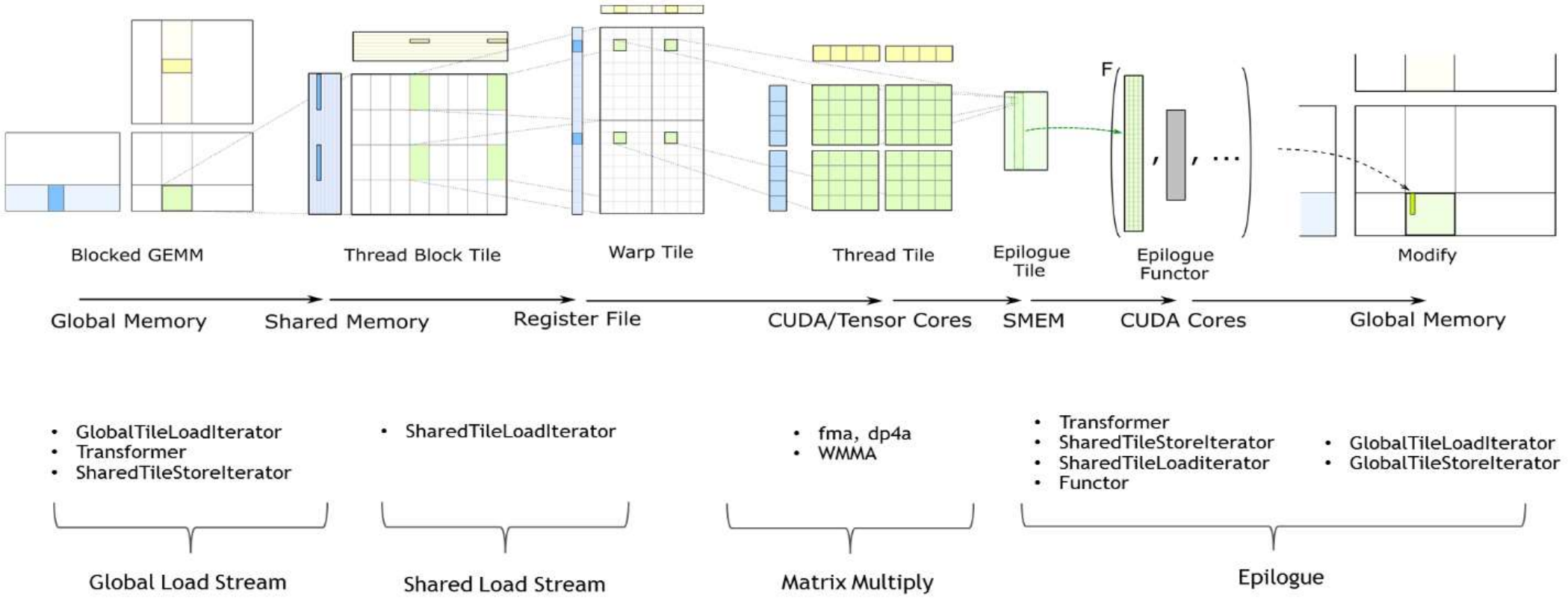


The GPU Devotes More Transistors to Data Processing



https://docs.nvidia.com/cuda/cuda-c-programming-guide/

GPU Memory Hierarchy



CUTLASS GEMM Structural Model

https://developer.nvidia.com/blog/cutlass-linear-algebra-cuda/

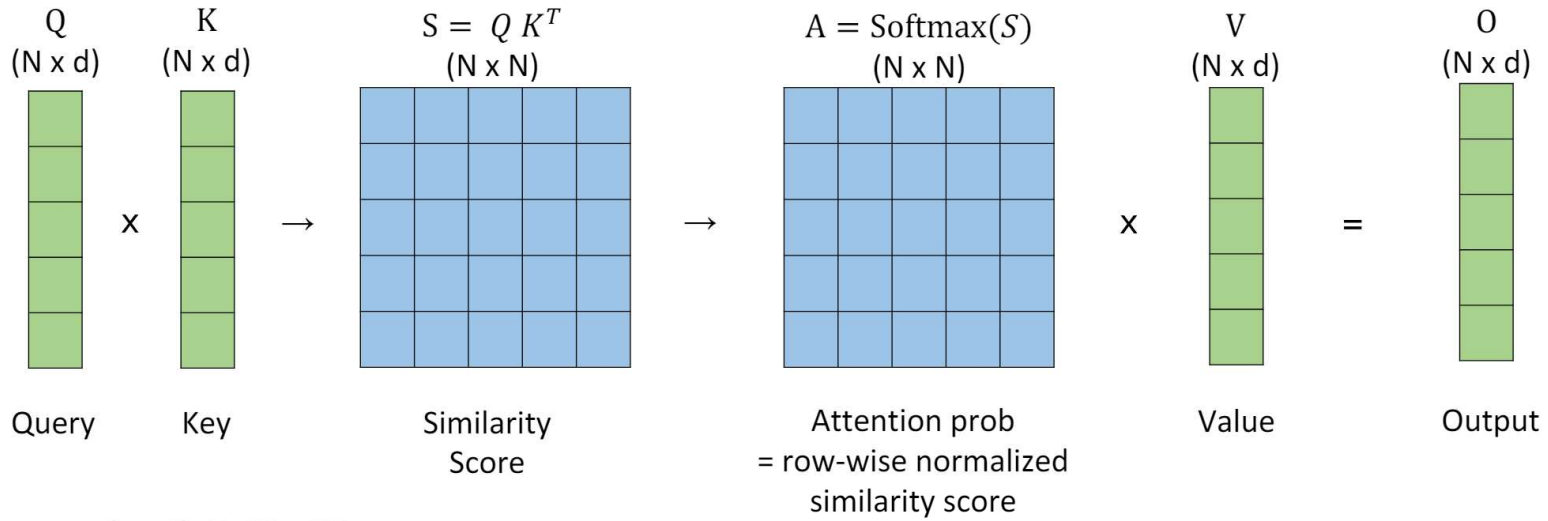
FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness

Tri Dao[†], Daniel Y. Fu[†], Stefano Ermon[†], Atri Rudra[‡], and Christopher Ré[†]

[†]Department of Computer Science, Stanford University
[‡]Department of Computer Science and Engineering, University at Buffalo, SUNY

{trid,danfu}@cs.stanford.edu, ermon@stanford.edu, atri@buffalo.edu,
chrismre@cs.stanford.edu

FlashAttention



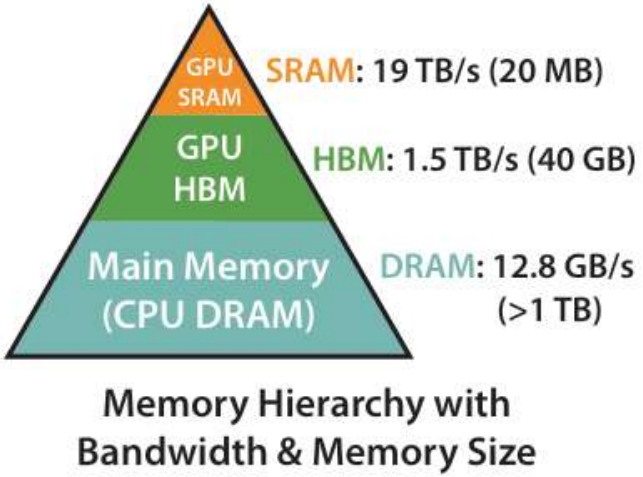
Typical sequence length N: 1K – 8K
Head dimension d: 64 – 128

$$\text{Softmax}([s_1, \cdots, s_N]) = \left[\frac{e^{s_1}}{\sum_i e^{s_i}}, \cdots, \frac{e^{s_N}}{\sum_i e^{s_i}}\right]$$

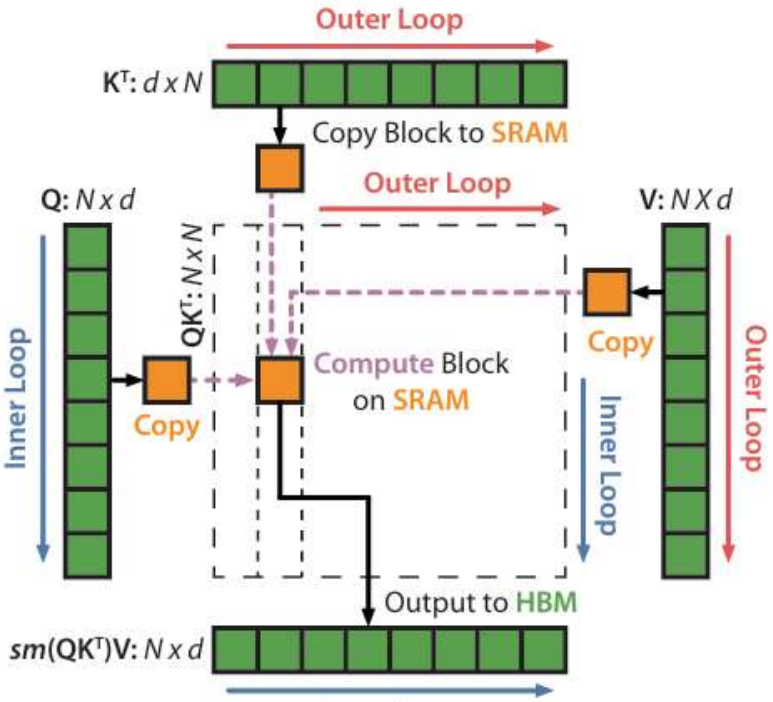
$$O = \text{Softmax}(QK^T)V$$

1: Load $\mathbf{Q}, \mathbf{K}$ by blocks from HBM, compute $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top$, write $\mathbf{S}$ to HBM.
2: Read $\mathbf{S}$ from HBM, compute $\mathbf{P} = \text{softmax}(\mathbf{S})$, write $\mathbf{P}$ to HBM.
3: Load $\mathbf{P}$ and $\mathbf{V}$ by blocks from HBM, compute $\mathbf{O} = \mathbf{P}\mathbf{V}$, write $\mathbf{O}$ to HBM.
4: Return $\mathbf{O}$.

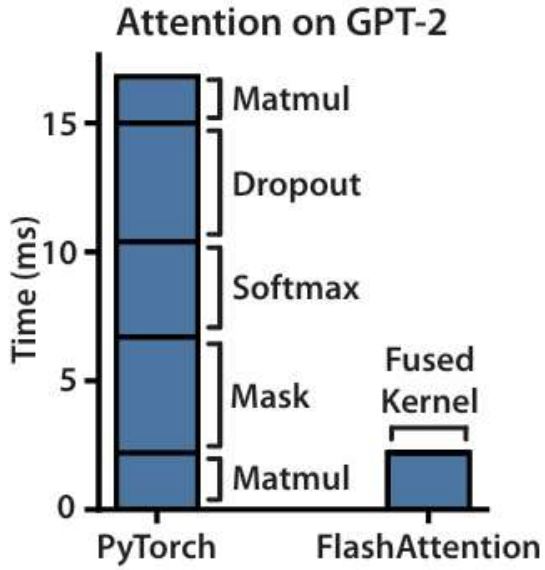
FlashAttention



Memory Hierarchy with Bandwidth & Memory Size



FlashAttention



FlashAttention

Algorithm 1 FlashAttention

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM, on-chip SRAM of size M.

Set block sizes $B_c = \lceil \frac{M}{4d} \rceil$, $B_r = \min\left(\lceil \frac{M}{4d} \rceil, d\right)$.

Initialize $\mathbf{O} = (0)_{N \times d} \in \mathbb{R}^{N \times d}$, $\ell = (0)_N \in \mathbb{R}^N$, $m = (-\infty)_N \in \mathbb{R}^N$ in HBM.

Divide $\mathbf{Q}$ into $T_r = \left\lceil \frac{N}{B_r} \right\rceil$ blocks $\mathbf{Q}_1, \ldots, \mathbf{Q}_{T_r}$ of size $B_r \times d$ each, and divide $\mathbf{K}, \mathbf{V}$ in to $T_c = \left\lceil \frac{N}{B_c} \right\rceil$ blocks $\mathbf{K}_1, \ldots, \mathbf{K}_{T_c}$ and $\mathbf{V}_1, \ldots, \mathbf{V}_{T_c}$, of size $B_c \times d$ each.

Divide $\mathbf{O}$ into T_r blocks $\mathbf{O}_i, \ldots, \mathbf{O}_{T_r}$ of size $B_r \times d$ each, divide ℓ into T_r blocks $\ell_i, \ldots, \ell_{T_r}$ of size B_r each, divide m into T_r blocks $m_1, \ldots, m_{T_r}$ of size B_r each.

for $1 \le j \le T_c$ **do**

  Load $\mathbf{K}_j, \mathbf{V}_j$ from HBM to on-chip SRAM.

  for $1 \le i \le T_r$ **do**

    Load $\mathbf{Q}_i, \mathbf{O}_i, \ell_i, m_i$ from HBM to on-chip SRAM.

    On chip, compute $\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_r \times B_c}$.

    On chip, compute $\tilde{m}_{ij} = \text{rowmax}(\mathbf{S}_{ij}) \in \mathbb{R}^{B_r}$, $\tilde{\mathbf{P}}_{ij} = \exp(\mathbf{S}_{ij} - \tilde{m}_{ij}) \in \mathbb{R}^{B_r \times B_c}$ (pointwise), $\tilde{\ell}_{ij} = \text{rowsum}(\tilde{\mathbf{P}}_{ij}) \in \mathbb{R}^{B_r}$.

    On chip, compute $m_i^{\text{new}} = \max(m_i, \tilde{m}_{ij}) \in \mathbb{R}^{B_r}$, $\ell_i^{\text{new}} = e^{m_i - m_i^{\text{new}}} \ell_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \tilde{\ell}_{ij} \in \mathbb{R}^{B_r}$.

    Write $\mathbf{O}_i \leftarrow \text{diag}(\ell_i^{\text{new}})^{-1}(\text{diag}(\ell_i) e^{m_i - m_i^{\text{new}}} \mathbf{O}_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \tilde{\mathbf{P}}_{ij} \mathbf{V}_j)$ to HBM.

    Write $\ell_i \leftarrow \ell_i^{\text{new}}$, $m_i \leftarrow m_i^{\text{new}}$ to HBM.

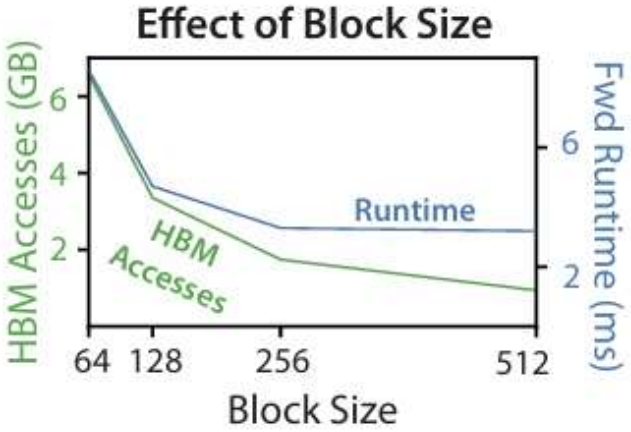
  end for

end for

Return $\mathbf{O}$.

FlashAttention

Attention	Standard	FLASHATTENTION
GFLOPs	66.6	75.2
HBM R/W (GB)	40.3	4.4
Runtime (ms)	41.7	7.3



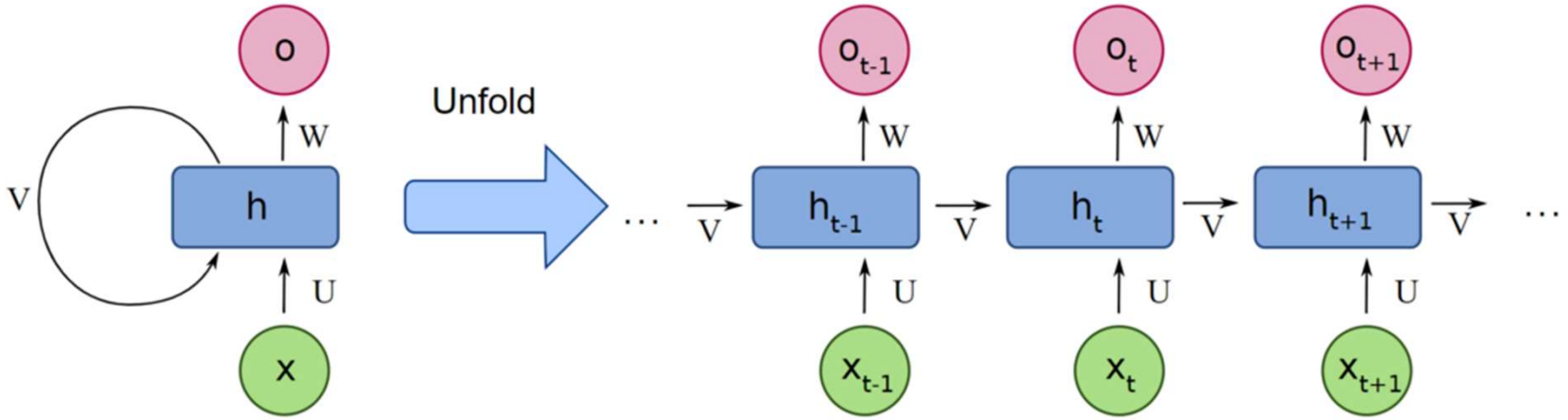
HBM access

FlashAttention: $\Theta(N^2d^2M^{-1})$

StandardAttention: $\Theta(Nd + N^2)$

d: 64/ 128
N: ~1024
M: ~100KB (A100)

Glance at RNN and Transformer



RNN:

$$h_t = \sigma_h(\mathbf{W}_h x_t + \mathbf{U}_h h_{t-1})$$

$$y_t = \sigma_y(\mathbf{W}_y h_t)$$

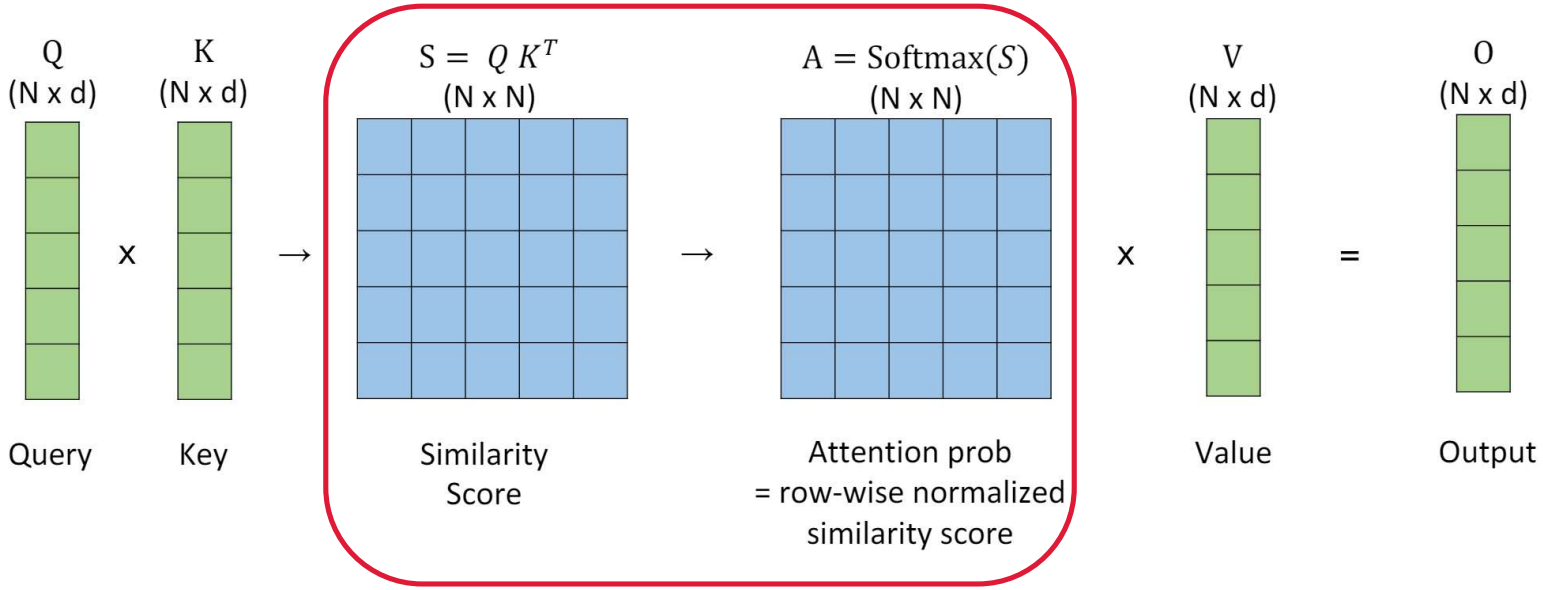
The recurrent formulation is not good for training!

SSM:

$$h_t = \overline{A} h_{t-1} + \overline{B} x_t$$

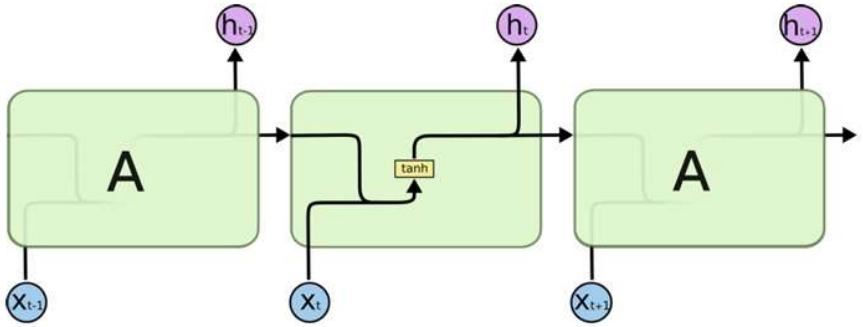
$$y_t = C h_t$$

Glance at RNN and Transformer



Attention scales quadratically in sequence length N.

Glance at RNN and Transformer

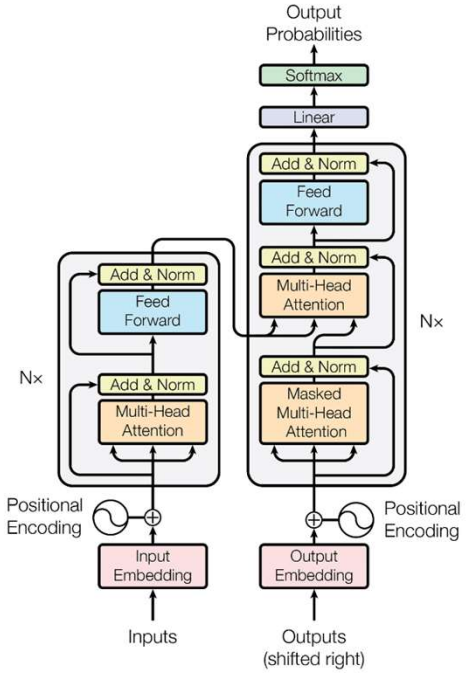


Natural autoregressive (causal) model

Linear-time training and inference

Bad parallelizable

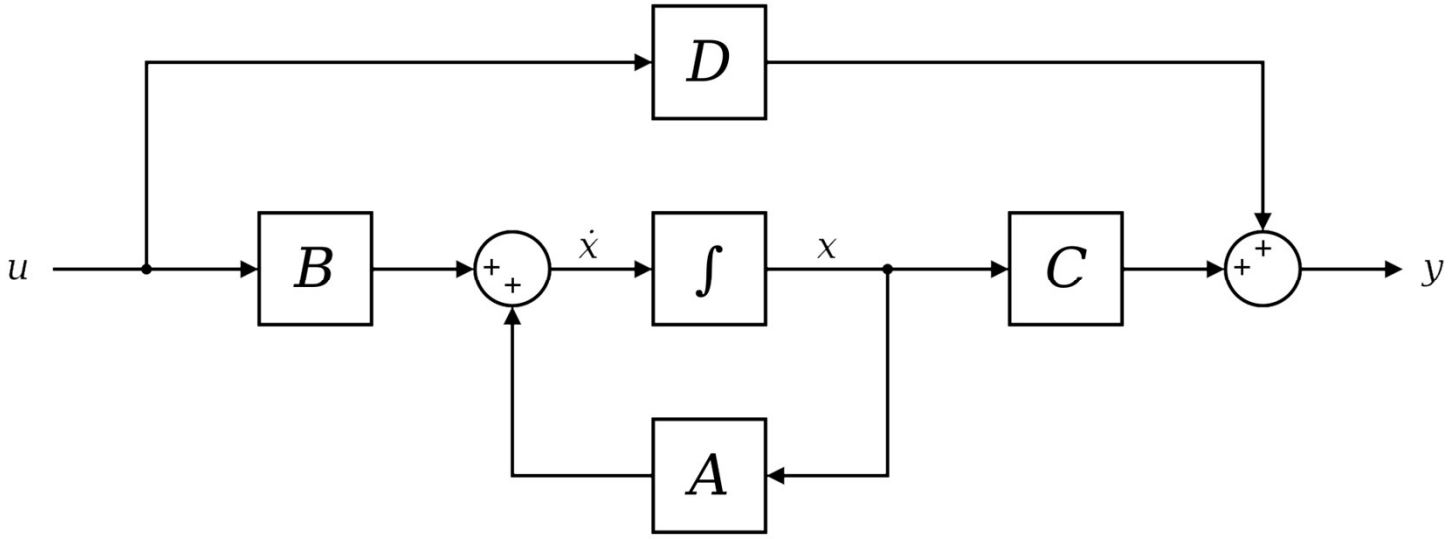
Vanishing gradients



Good performance
Good parallelizable

Quadratic-time training

SSM



$$h'(t) = \mathbf{A}h(t) + \mathbf{B}u(t)$$

$$y(t) = \mathbf{C}h(t) + \mathbf{D}u(t)$$

$h(t) \in \mathbb{R}^n$ state variables

$u(t) \in \mathbb{R}^m$ state inputs

$y(t) \in \mathbb{R}^p$ outputs

$\mathbf{A} \in \mathbb{R}^{m \times n}$ state matrix

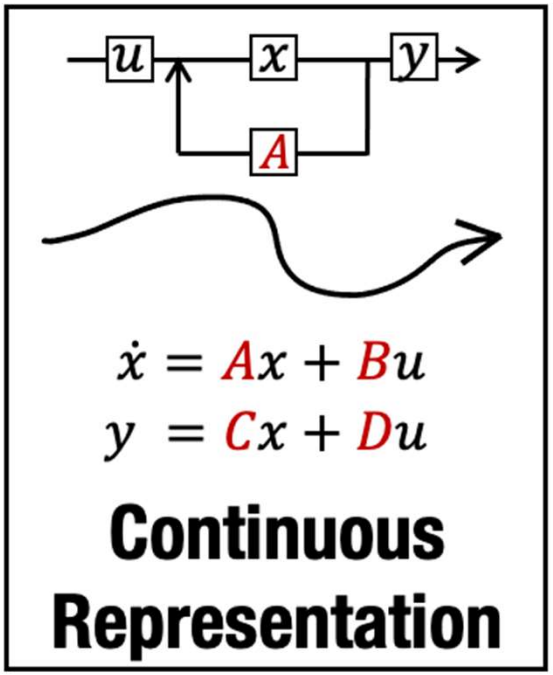
$\mathbf{B} \in \mathbb{R}^{n \times m}$ control matrix

$\mathbf{C} \in \mathbb{R}^{p \times n}$ output matrix

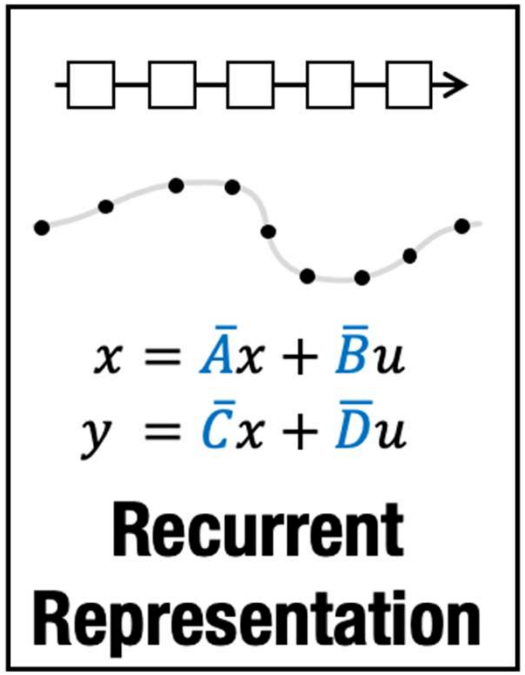
$\mathbf{D} \in \mathbb{R}^{p \times m}$ command matrix

Parameters are constant (invariant) through time (LTI system).

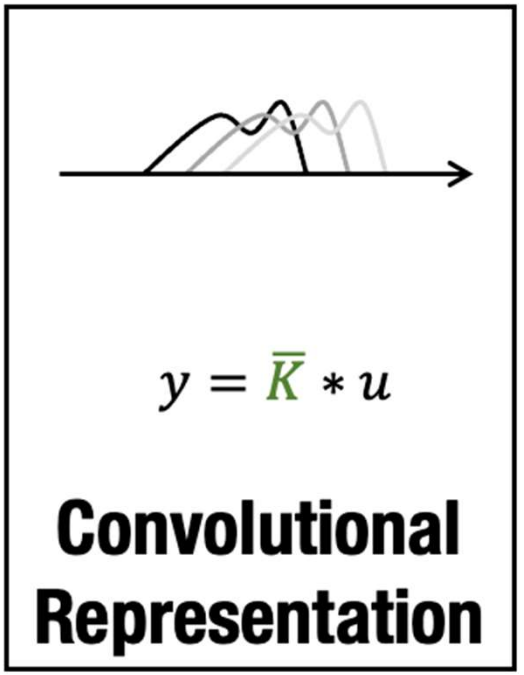
Selective State Spaces



Discretize



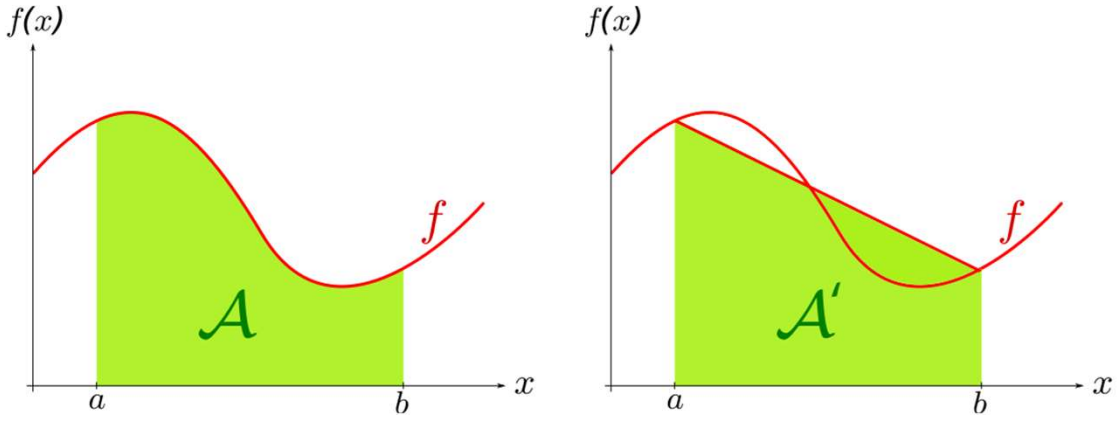
Unroll



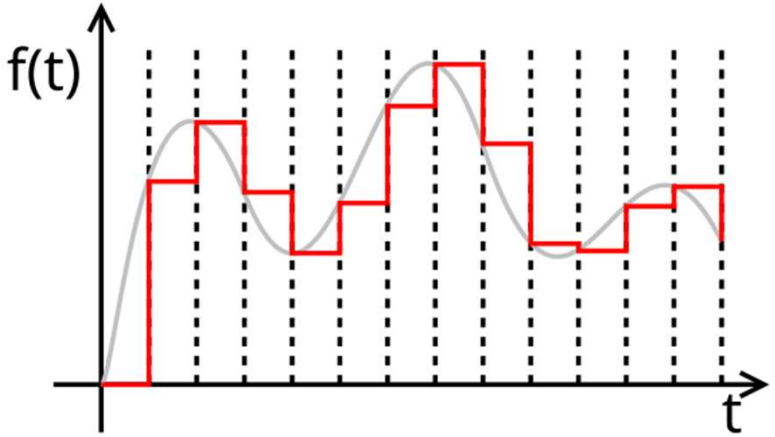
Efficient autoregressive computation

Efficient parallelizable computation

Selective State Spaces



Bilinear discretization



Zero-order hold (ZOH) discretization

Discretization	Bilinear	ZOH
Recurrence	$\bar{\mathbf{A}} = (\mathbf{I} - \frac{\Delta}{2}\mathbf{A})^{-1}(\mathbf{I} + \frac{\Delta}{2}\mathbf{A})$ $\bar{\mathbf{B}} = (\mathbf{I} - \frac{\Delta}{2}\mathbf{A})^{-1}\Delta\mathbf{B}$ $\bar{\mathbf{C}} = \mathbf{C}$	$\bar{\mathbf{A}} = e^{\mathbf{A}\Delta}$ $\bar{\mathbf{B}} = (\bar{\mathbf{A}} - I)\mathbf{A}^{-1}\mathbf{B}$ $\bar{\mathbf{C}} = \mathbf{C}$
Convolution	$\bar{\mathbf{K}}_k = (\bar{\mathbf{C}}\bar{\mathbf{B}}, \bar{\mathbf{C}}\bar{\mathbf{A}}\bar{\mathbf{B}}, \ldots, \bar{\mathbf{C}}\bar{\mathbf{A}}^k\bar{\mathbf{B}}$ $)$	$\bar{\mathbf{K}} = (\ \mathbf{C}e^{\mathbf{A}\cdot k\Delta}(e^{\mathbf{A}\Delta} -$ $I)\mathbf{A}^{-1}\mathbf{B}\ )_{0\leq k<L}$

Selective State Spaces

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t$$
$$y_t = Ch_t$$

ZOH:

$$\overline{A} = \exp(\Delta A)$$

$$\overline{B} = (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B$$

$$h_0 = \bar{B}x_0$$
$$y_0 = Ch_0 = C\bar{B}x_0$$

$$h_1 = \bar{A}h_0 + \bar{B}x_1 = \bar{A}\bar{B}x_0 + \bar{B}x_1$$
$$y_1 = Ch_1 = C(\bar{A}\bar{B}x_0 + \bar{B}x_1) = C\bar{A}\bar{B}x_0 + C\bar{B}x_1$$

$$h_2 = \bar{A}h_1 + \bar{B}x_2 = \bar{A}(\bar{A}\bar{B}x_0 + \bar{B}x_1) + \bar{B}x_2 = \bar{A}^2\bar{B}x_0 + \bar{A}\bar{B}x_1 + \bar{B}x_2$$
$$y_2 = Ch_2 = C(\bar{A}^2\bar{B}x_0 + \bar{A}\bar{B}x_1 + \bar{B}x_2) = C\bar{A}^2\bar{B}x_0 + C\bar{A}\bar{B}x_1 + C\bar{B}x_2$$

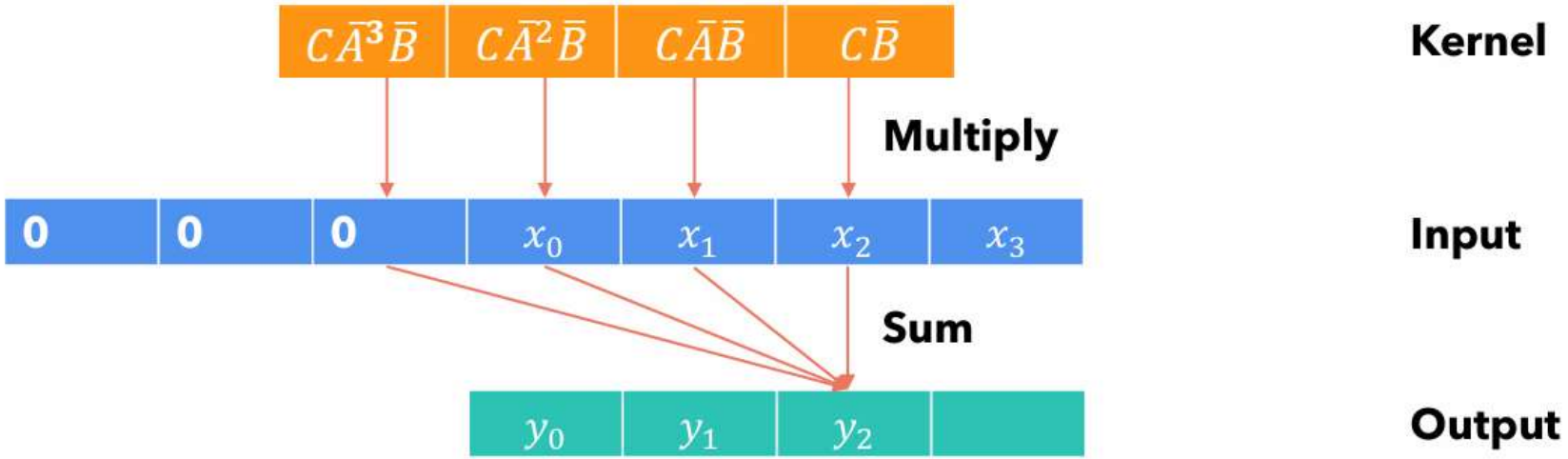
$$y_k = C\bar{A}^k\bar{B}x_0 + C\bar{A}^{k-1}\bar{B}x_1 + \cdots + C\bar{A}\bar{B}x_{k-1} + C\bar{B}x_k$$

Selective State Spaces

$$y_k = C\bar{A}^k\bar{B}x_0 + C\bar{A}^{k-1}\bar{B}x_1 + \cdots + C\bar{A}\bar{B}x_{k-1} + C\bar{B}x_k$$

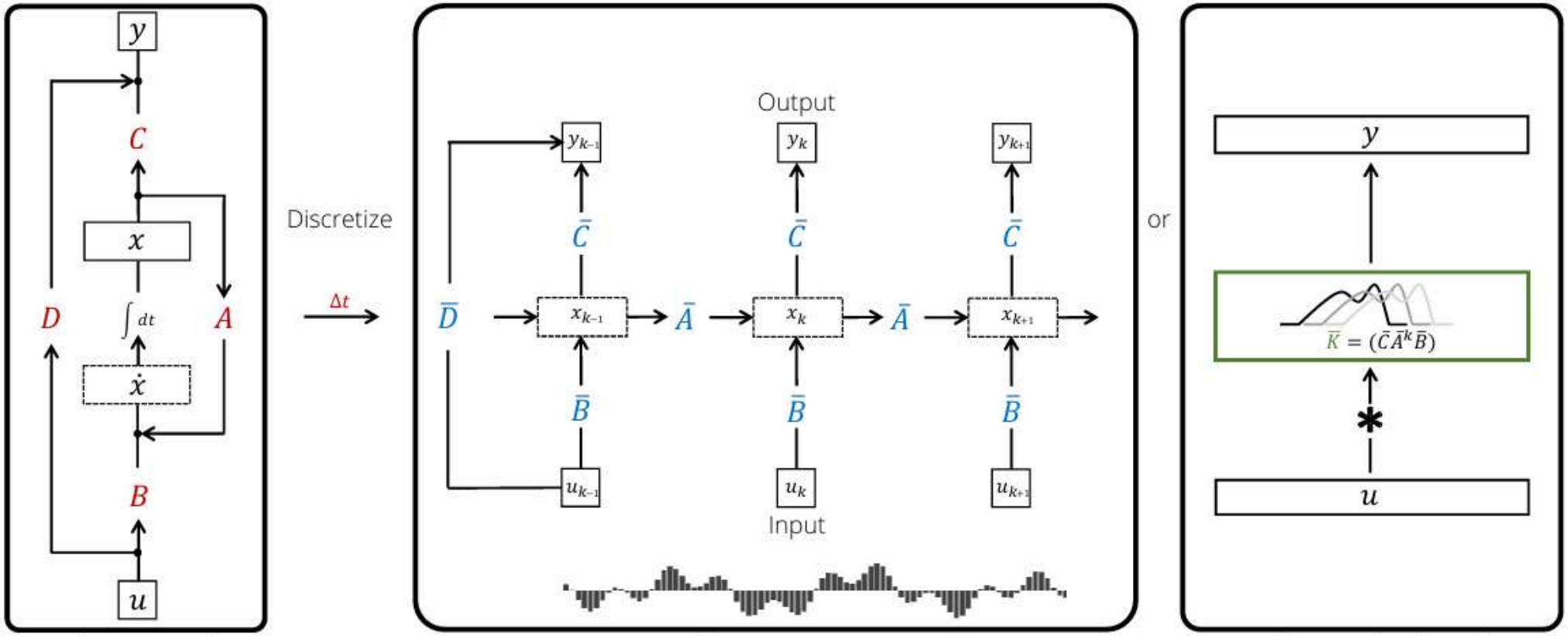
$$\overline{\boldsymbol{K}} = (C\overline{\boldsymbol{B}}, C\overline{\boldsymbol{AB}}, \ldots, C\overline{\boldsymbol{A}}^{\boldsymbol{k}}\overline{\boldsymbol{B}}, \ldots)$$

$$y = x * \overline{\boldsymbol{K}}$$



$$y_2 = C\bar{A}^2\bar{B}x_0 + C\bar{A}\bar{B}x_1 + C\bar{B}x_2$$

Selective State Spaces



Continuous-time

- ✓ *continuous data*
- ✓ *irregular sampling*

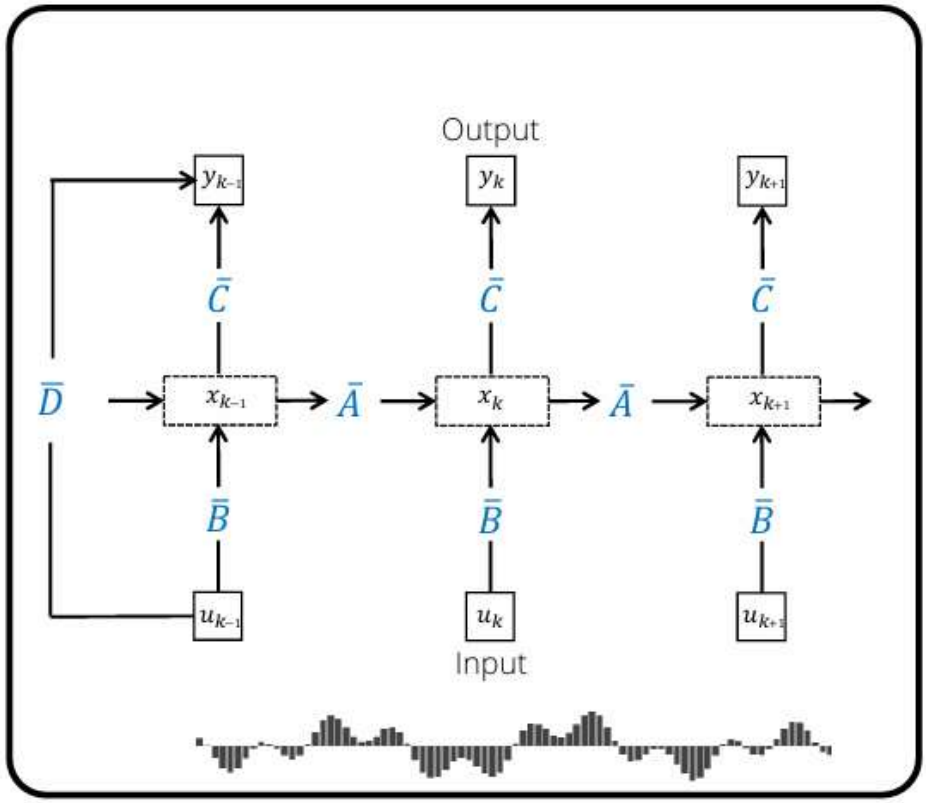
Recurrent

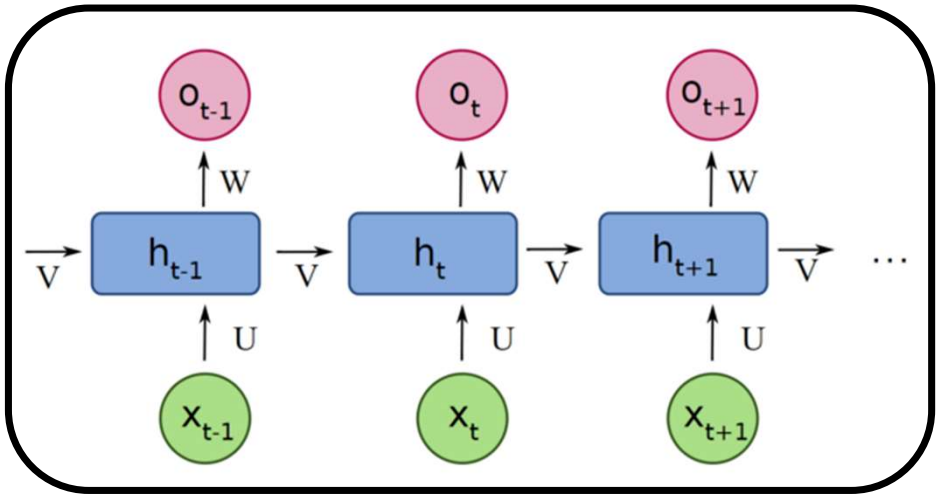
- ✓ *unbounded context*
- ✓ *efficient inference*

Convolutional

- ✓ *local information*
- ✓ *parallelizable training*

SSM and RNN

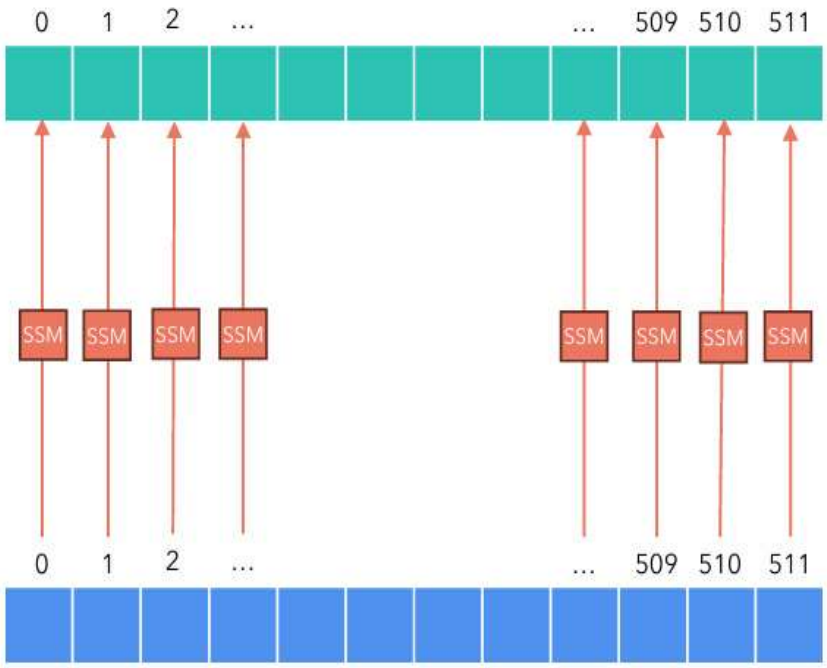


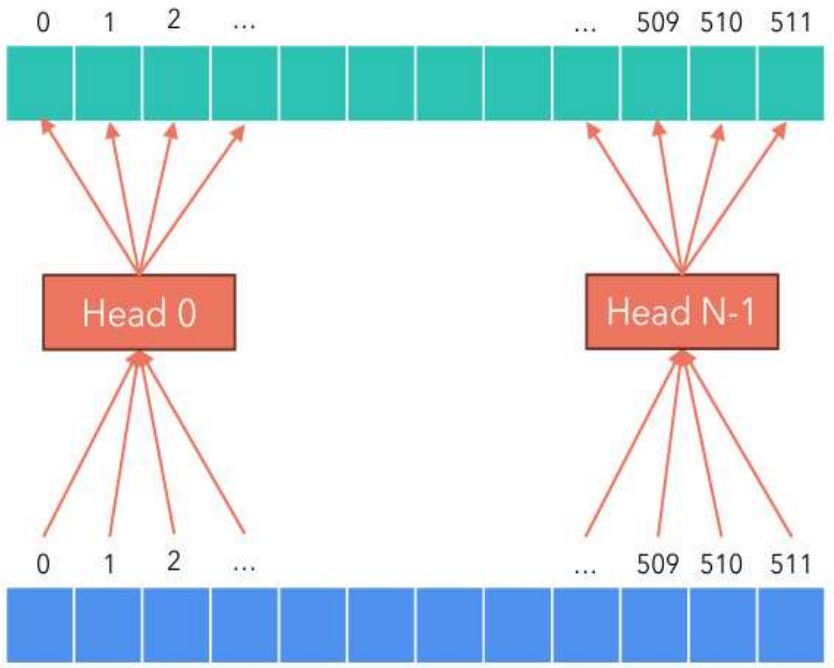


$$h_t = \overline{A}h_{t-1} + \overline{B}x_t$$
$$y_t = Ch_t$$

$$h_t = \sigma_h(\mathbf{W}_h x_t + \mathbf{U}_h h_{t-1})$$
$$y_t = \sigma_y(\mathbf{W}_y h_t)$$

SSM and Attention





SSM: Each dimension is managed by an **independent** state space model.

Transformer: Each group of dimensions is managed by a different head of multi-head attention.

Selective State Spaces

$$h'(t) = \mathbf{A}h(t) + \mathbf{B}u(t)$$
$$y(t) = \mathbf{C}h(t) + \mathbf{D}u(t)$$

$\mathbf{A} \in \mathbb{R}^{m \times n}$ state matrix

$\mathbf{B} \in \mathbb{R}^{n \times m}$ control matrix

Discretize

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t$$
$$y_t = Ch_t$$

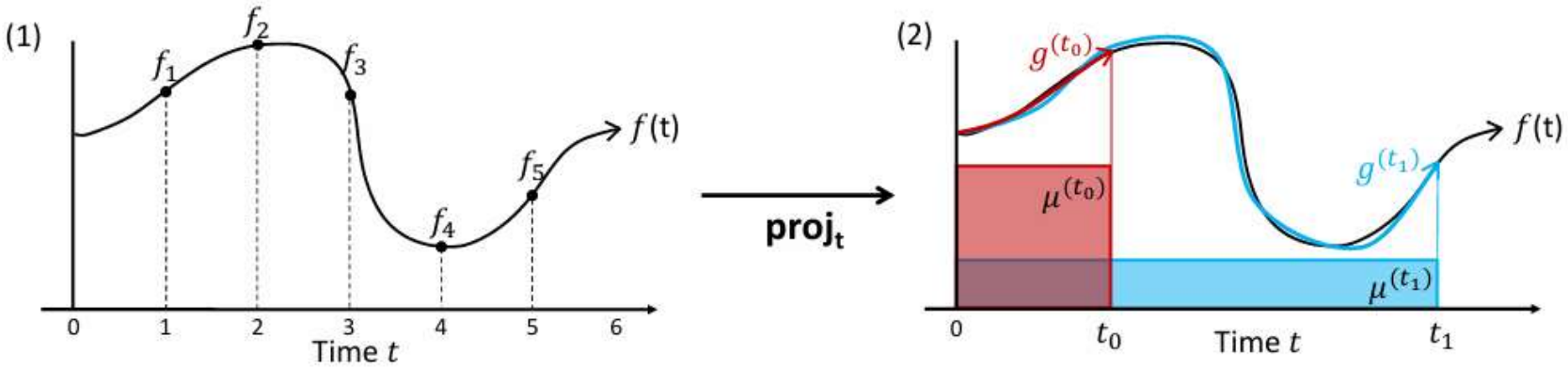
ZOH:

$$\overline{A} = \exp(\Delta A)$$

$$\overline{B} = (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B$$

$\overline{A}$ captures information from the previous state to build the new state!

HiPPO



Online Function Approximation

Gu A, Dao T, Ermon S, et al. Hippo: Recurrent memory with optimal polynomial projections, 2020.

Gu A, Goel K, Ré C. Efficiently modeling long sequences with structured state spaces, 2021.

HiPPO

A set of **complete polynomials** can approximate an arbitrary function by **linear combination**.

- Fourier series
- Legendre polynomials

Legendre polynomials:

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}[(x^2-1)^n]$$

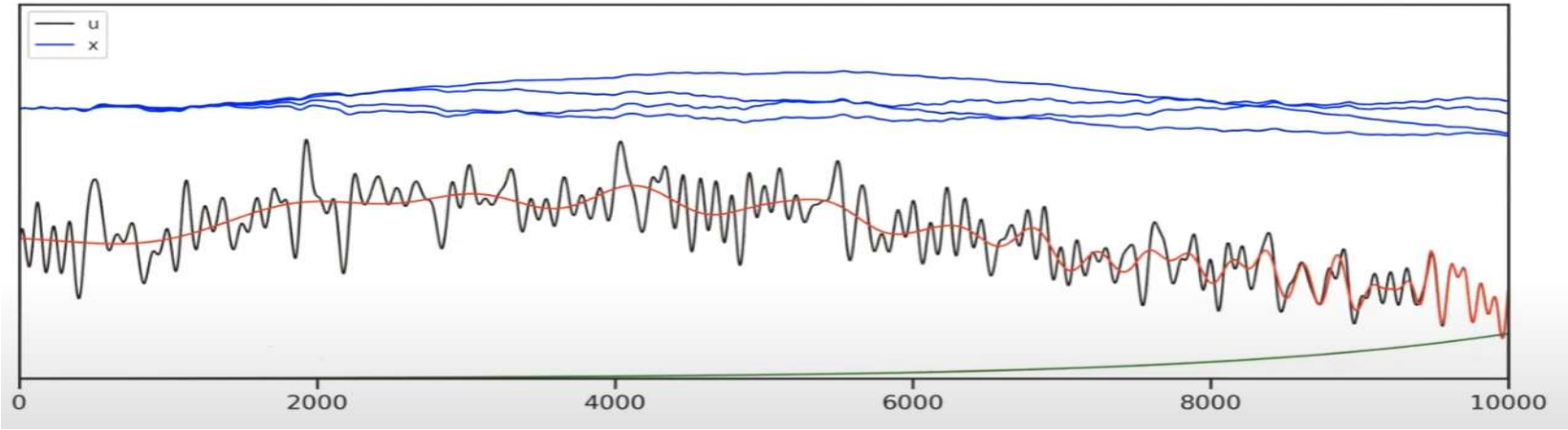
The Fourier transformation allows us to decompose a signal into sinusoidal functions

https://en.wikipedia.org/wiki/Fourier_transform

HiPPO

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t$$
$$y_t = Ch_t$$
$$\overline{A} = \exp(\Delta A)$$
$$\overline{B} = (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B$$

A matrix can build in such a way that it approximates all the input signal seen so far into a vector of coefficients by Legendre polynomials.



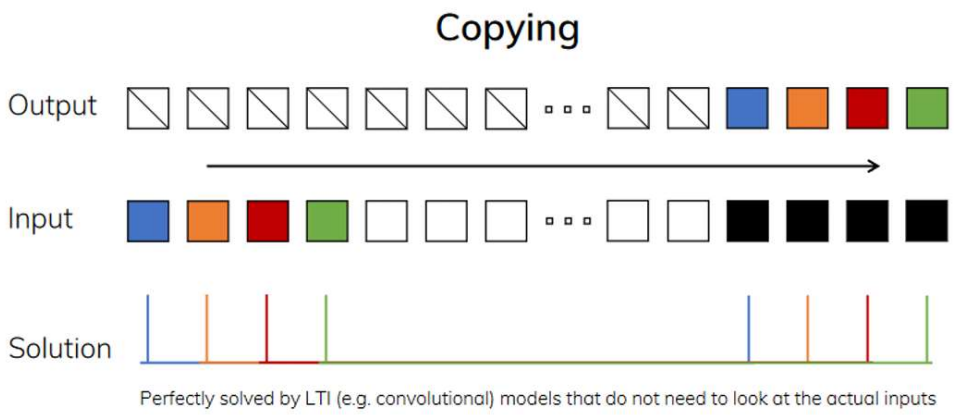
As opposed to traditional function approximation instead of constructing all signals perfectly, new signals are constructed exactly, and old signals decay exponentially, so being able to h(t) is able to capture recent tokens.

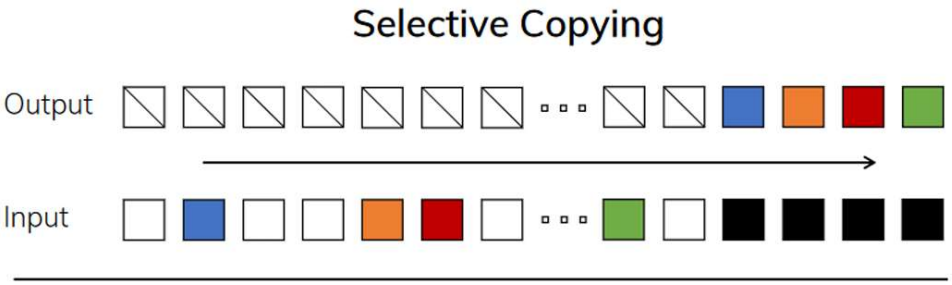
Mamba: Linear-Time Sequence Modeling with Selective State Spaces

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

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

Mamba



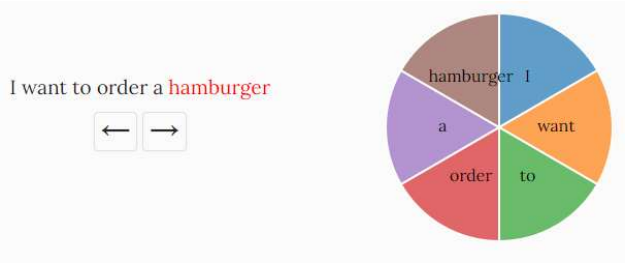


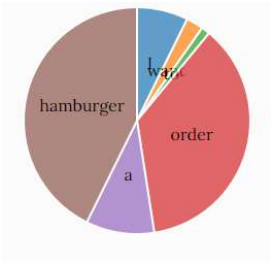
Intuition: rewrite the input one token at a time, but time-shifted.

This **can** be performed by a vanilla SSM, and the time delay can be learnt by a convolution.

Intuition: given a comment on Twitter, rewrite the comment by removing all the bad words (the white tokens).

This **cannot** be performed by a vanilla SSM, because it requires content-aware reasoning, which vanilla SSM cannot do because they are time invariant (which means the parameters A, B, C, D are the same for every token it generates).





Mamba: The Easy Way:https://jackcook.com/2024/02/23/mamba.html

Mamba

Algorithm 1 SSM (S4)

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

$$h_t = \overline{A} h_{t-1} + \overline{B} x_t$$
$$y_t = C h_t$$

The $\bar{A}$ matrix also depends on the input, through Δ

Algorithm 2 SSM + Selection (S6)

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

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

B: Batch size
L: Sequence length
D: Size of the input vector
N: Size of the hidden state *h*

Mamba

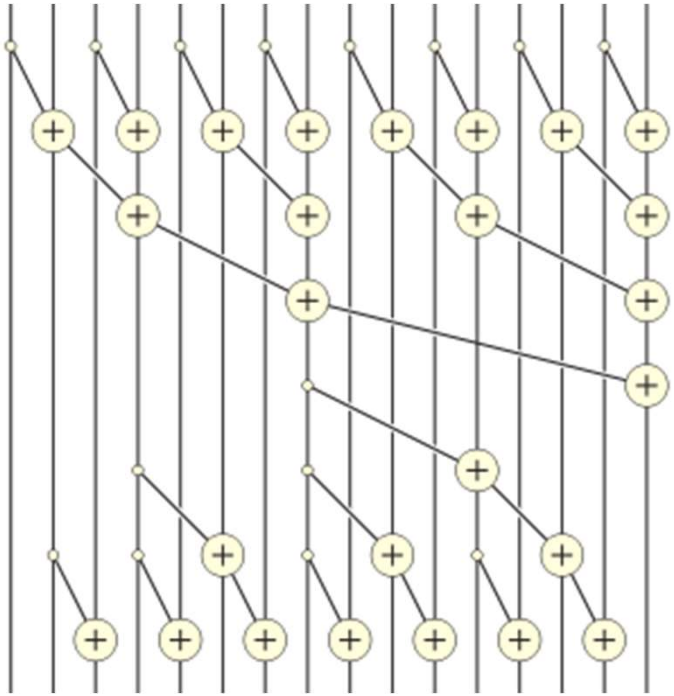
Algorithm 2 SSM + Selection (S6)

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

$$\overline{\boldsymbol{K}} = (C\overline{B}, C\overline{AB}, \dots, C\overline{A}^{k}\overline{B}, \dots)$$
$$y = x * \overline{\boldsymbol{K}}$$

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t$$
$$y_t = Ch_t$$

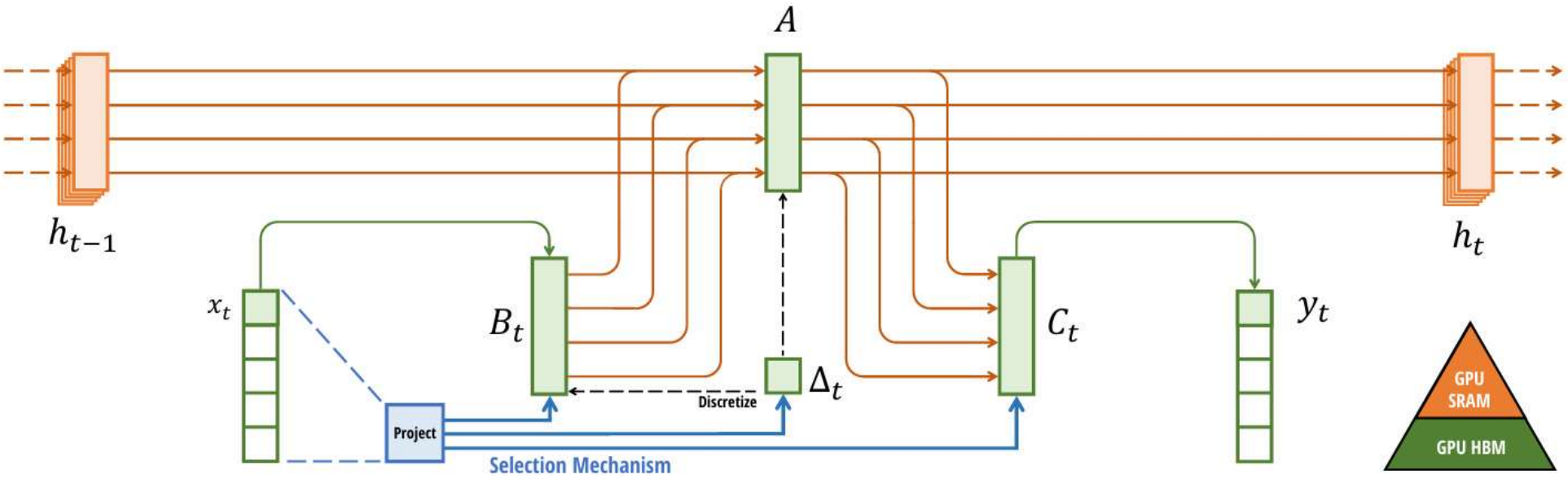
Parallel prefix-sum



$$y_t = y_{t-1} + x_t$$

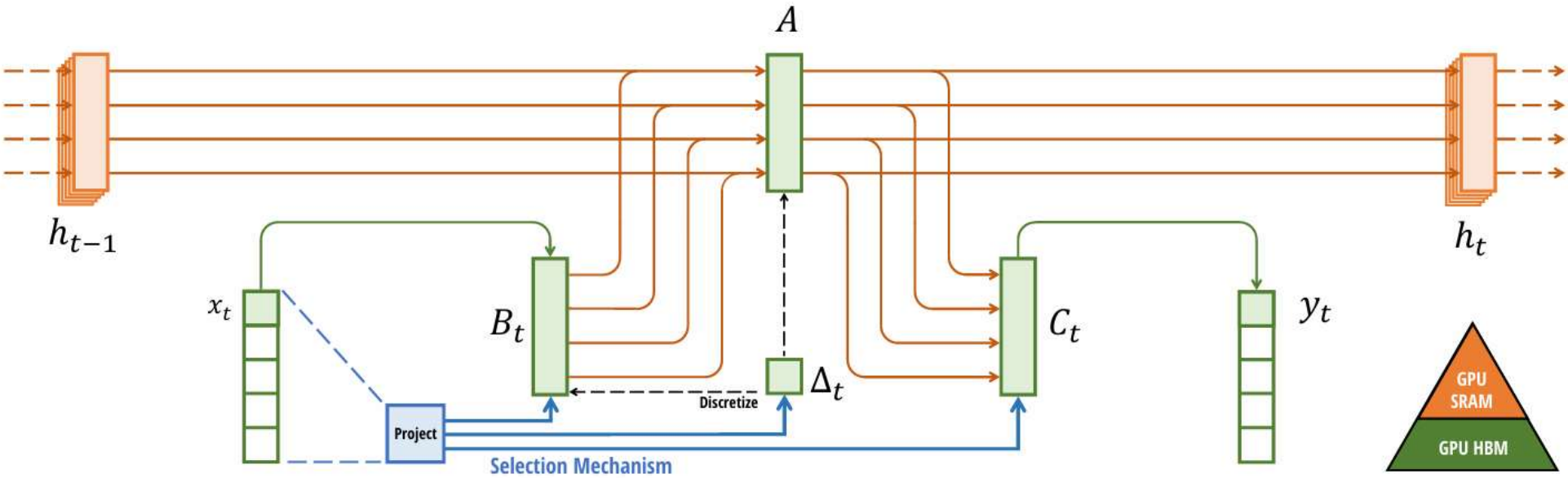
https://en.wikipedia.org/wiki/Prefix_sum

Mamba



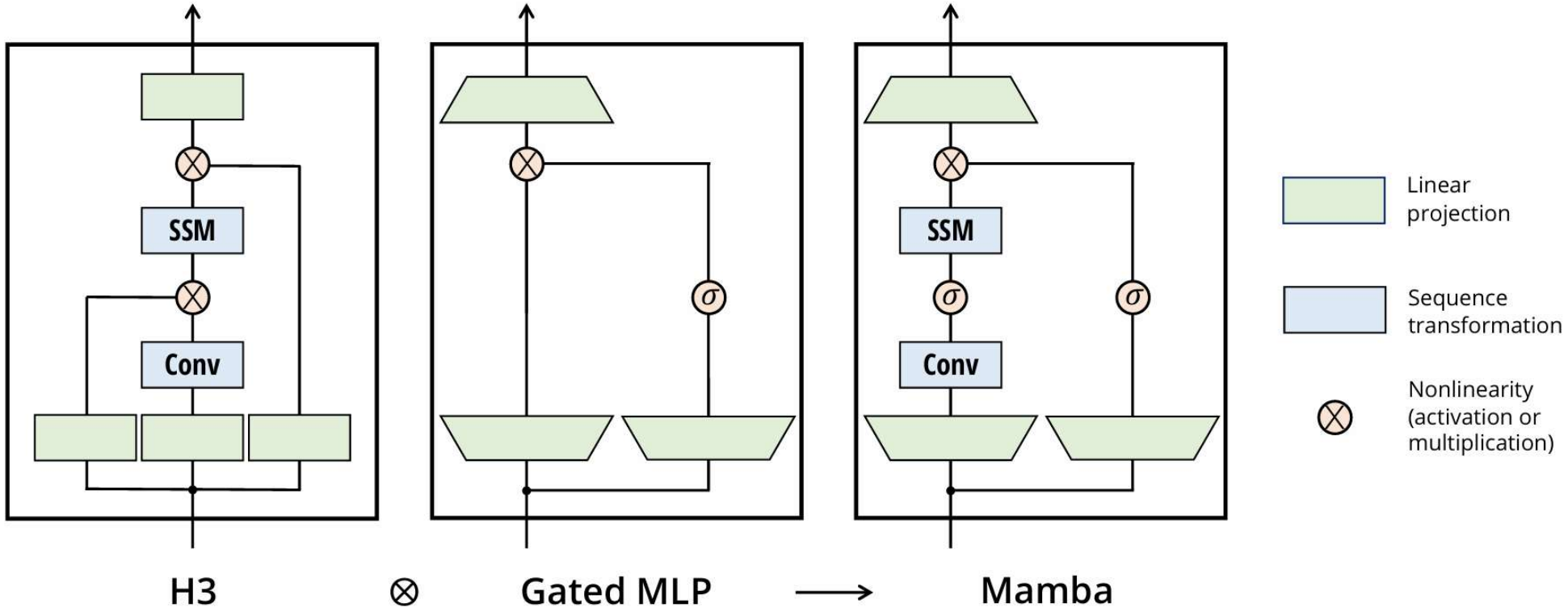
1. We read in $O(BLD + DN)$ bytes of memory $(\Delta, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ from slow HBM to fast SRAM.
2. We discretize to produce $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}$ of size (B, L, D, N) in SRAM.
3. We perform a parallel associative scan, yielding intermediate states of size (B, L, D, N) in SRAM.
4. We multiply and sum with $\boldsymbol{C}$, producing outputs of size (B, L, D) and write it to HBM.

Mamba



- ✓ Efficiency: parallelizable training + fast inference
- ✓ Performance: matches Transformers on LM
- ✓ Long Context: improves up to million-length sequences

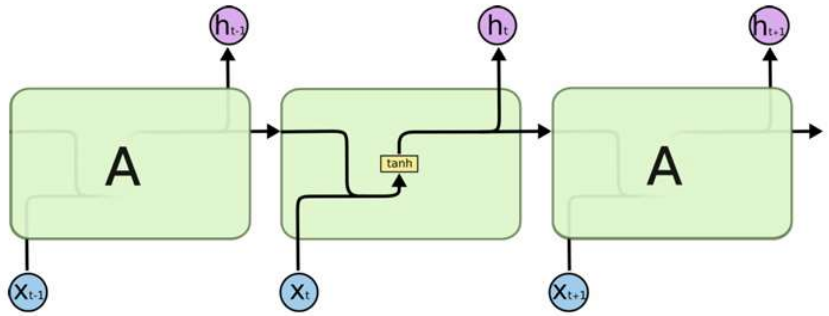
Mamba



Hungry Hungry Hippo

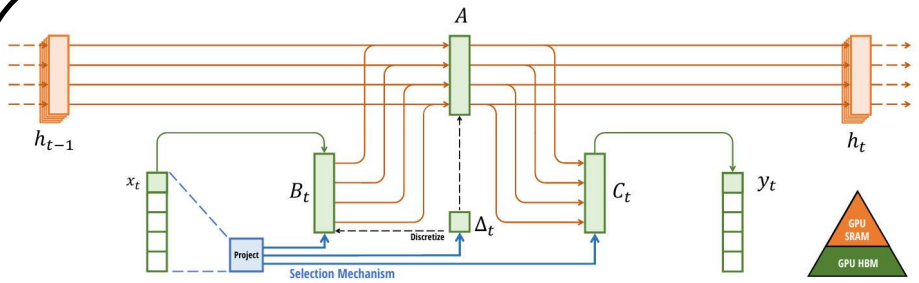
Summary

What different between RNN and Mamba?



$$h_t = \sigma_h(\mathbf{W}_h x_t + \mathbf{U}_h h_{t-1})$$

$$y_t = \sigma_y(\mathbf{W}_y h_t)$$



1. Removed the nonlinear layer and able to parallelize the computation;
2. Design Matrix ***A*** to Reduce Hidden State Obliviousness;
3. Each dimension corresponds to an SSM module.

Mamba for CV

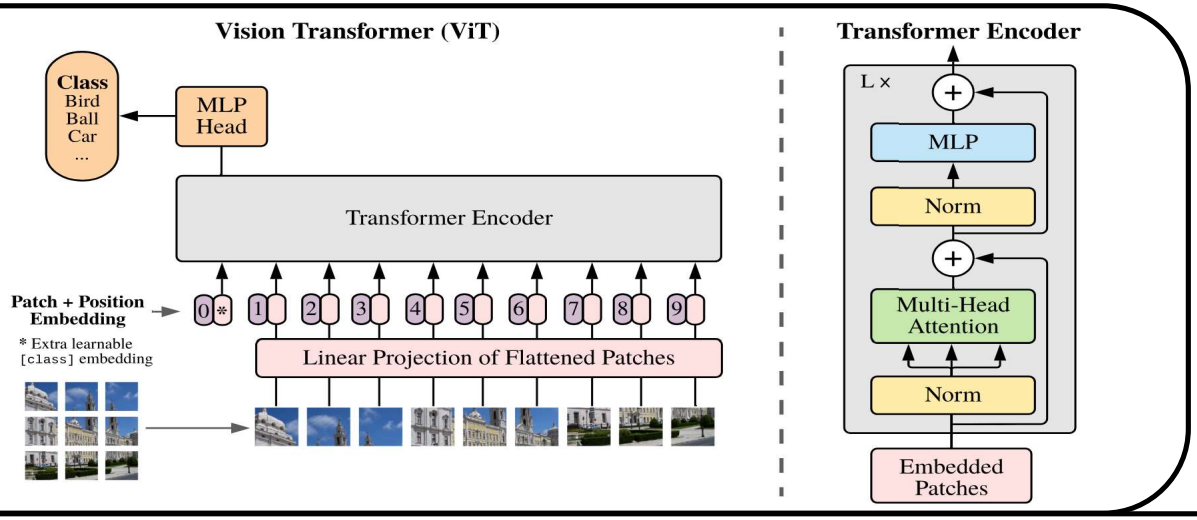
- LFMamba: Light Field Image Super-Resolution with State Space Model [arxiv]
- Voxel Mamba: Group-Free State Space Models for Point Cloud based 3D Object Detection [arxiv] [code]
- PyramidMamba: Rethinking Pyramid Feature Fusion with Selective Space State Model for Semantic Segmentation of Remote Sensing Imagery [arxiv] [code]
- Q-Mamba: On First Exploration of Vision Mamba for Image Quality Assessment [arxiv]
- PixMamba: Leveraging State Space Models in a Dual-Level Architecture for Underwater Image Enhancement [arxiv] [code]
- Towards Evaluating the Robustness of Visual State Space Models [arxiv] [code]
- DualMamba: A Lightweight Spectral-Spatial Mamba-Convolution Network for Hyperspectral Image Classification [arxiv]
- Autoregressive Pretraining with Mamba in Vision [arxiv] [code]
- MambaDepth: Enhancing Long-range Dependency for Self-Supervised Fine-Structured Monocular Depth Estimation [arxiv] [code]
- Efficient 3D Shape Generation via Diffusion Mamba with Bidirectional SSMs [arxiv]
- MHS-VM: Multi-Head Scanning in Parallel Subspaces for Vision Mamba [arxiv] [code]
- HDMba: Hyperspectral Remote Sensing Imagery Dehazing with State Space Model [arxiv] [code]
- Mamba YOLO: SSMs-Based YOLO For Object Detection [arxiv] [code]
- MVGamba: Unify 3D Content Generation as State Space Sequence Modeling [arxiv]
- RoboMamba: Multimodal State Space Model for Efficient Robot Reasoning and Manipulation [arxiv] [code]
- GrootVL: Tree Topology is All You Need in State Space Model [arxiv] [code]
- CDMamba: Remote Sensing Image Change Detection with Mamba [arxiv] [code]
- LLEMamba: Low-Light Enhancement via Relighting-Guided Mamba with Deep Unfolding Network [arxiv]
- Dimba: Transformer-Mamba Diffusion Models [arxiv] [code]
- S4Fusion: Saliency-aware Selective State Space Model for Infrared Visible Image Fusion [arxiv]
- DeMamba: AI-Generated Video Detection on Million-Scale GenVideo Benchmark [arxiv] [code]
- FourierMamba: Fourier Learning Integration with State Space Models for Image Deraining [arxiv]
- Vim-F: Visual State Space Model Benefiting from Learning in the Frequency Domain [arxiv] [code]
- MambaLLIE: Implicit Retinex-Aware Low Light Enhancement with Global-then-Local State Space [arxiv] [code]
- Image Deraining with Frequency-Enhanced State Space Model [arxiv]
- Demystify Mamba in Vision: A Linear Attention Perspective [arxiv] [code]
- MambaVC: Learned Visual Compression with Selective State Spaces [arxiv]
- PoinTramba: A Hybrid Transformer-Mamba Framework for Point Cloud Analysis [arxiv] [code]
- Meteor: Mamba-based Traversal of Rationale for Large Language and Vision Models [arxiv] [code]
- Multi-Scale VMamba: Hierarchy in Hierarchy Visual State Space Model [arxiv] [code]
- DiM: Diffusion Mamba for Efficient High-Resolution Image Synthesis [arxiv]
- MAMBA4D: Efficient Long-Sequence Point Cloud Video Understanding with Disentangled Spatial-Temporal State Space Models [arxiv]
- SpikeMba: Multi-Modal Spiking Saliency Mamba for Temporal Video Grounding [arxiv]
- MambaMixer: Efficient Selective State Space Models with Dual Token and Channel Selection [arxiv] [code]
- Aggregating Local and Global Features via Selective State Spaces Model for Efficient Image Deblurring [arxiv]
- HARMamba: Efficient Wearable Sensor Human Activity Recognition Based on Bidirectional Selective SSM [arxiv]
- RSMamba: Remote Sensing Image Classification with State Space Model [arxiv] [code]
- Gamba: Marry Gaussian Splatting with Mamba for single view 3D reconstruction [arxiv]
- Integrating Mamba Sequence Model and Hierarchical Upsampling Network for Accurate Semantic Segmentation of Multiple Sclerosis Legion [arxiv]
- PlainMamba: Improving Non-Hierarchical Mamba in Visual Recognition [arxiv] [code]
- ReMamber: Referring Image Segmentation with Mamba Twister [arxiv]
- VMRNN: Integrating Vision Mamba and LSTM for Efficient and Accurate Spatiotemporal Forecasting [arxiv] [code]
- SiMBA: Simplified Mamba-based Architecture for Vision and Multivariate Time series [arxiv] [code]
- Cobra: Extending Mamba to Multi-Modal Large Language Model for Efficient Inference [arxiv] [code]
- VL-Mamba: Exploring State Space Models for Multimodal Learning [arxiv]
- ZigMa: Zigzag Mamba Diffusion Model [arxiv] [code]
- VmambaIR: Visual State Space Model for Image Restoration [arxiv] [code]
- LocalMamba: Visual State Space Model with Windowed Selective Scan [arxiv] [code]
- MambaTalk: Efficient Holistic Gesture Synthesis with Selective State Space Models [arxiv]
- Video Mamba Suite: State Space Model as a Versatile Alternative for Video Understanding [arxiv] [code]
- Motion Mamba: Efficient and Long Sequence Motion Generation with Hierarchical and Bidirectional Selective SSM [arxiv] [code]
- Point Mamba: A Novel Point Cloud Backbone Based on State Space Model with Octree-Based Ordering Strategy [arxiv] [code]
- VideoMamba: State Space Model for Efficient Video Understanding [arxiv] [code]
- MiM-ISTD: Mamba-in-Mamba for Efficient Infrared Small Target Detection [arxiv] [code]
- Point Could Mamba: Point Cloud Learning via State Space Model [arxiv] [code]
- Res-VMamba: Fine-Grained Food Category Visual Classification Using Selective State Space Models with Deep Residual Learning [arxiv] [code]
- MambaIR: A Simple Baseline for Image Restoration with State-Space Model [arxiv] [code]
- Pan-Mamba: Effective pan-sharpening with State Space Model [arxiv] [code]
- PointMamba: A Simple State Space Model for Point Cloud Analysis [arxiv] [code]
- Scalable Diffusion Models with State Space Backbone [arxiv] [code]
- Mamba-ND: Selective State Space Modeling for Multi-Dimensional Data [arxiv]
- Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model [arxiv] [code]
- VMamba: Visual State Space Model [arxiv] [code]

100+ papers on the use of Mamba in CV

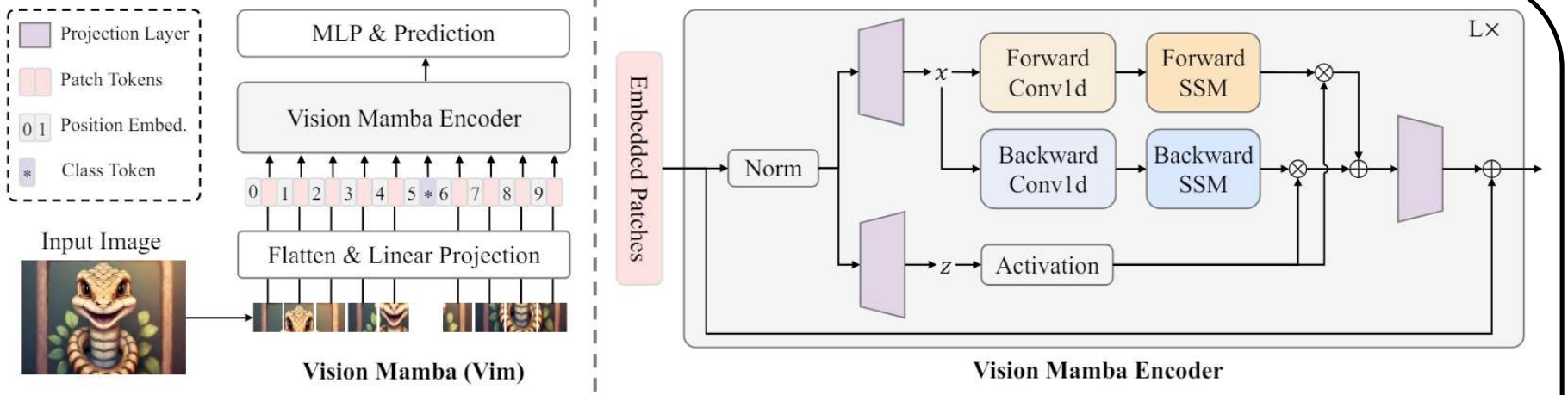
https://github.com/ReaFly/Awesome-Vision-Mamba

Mamba for CV

ViT



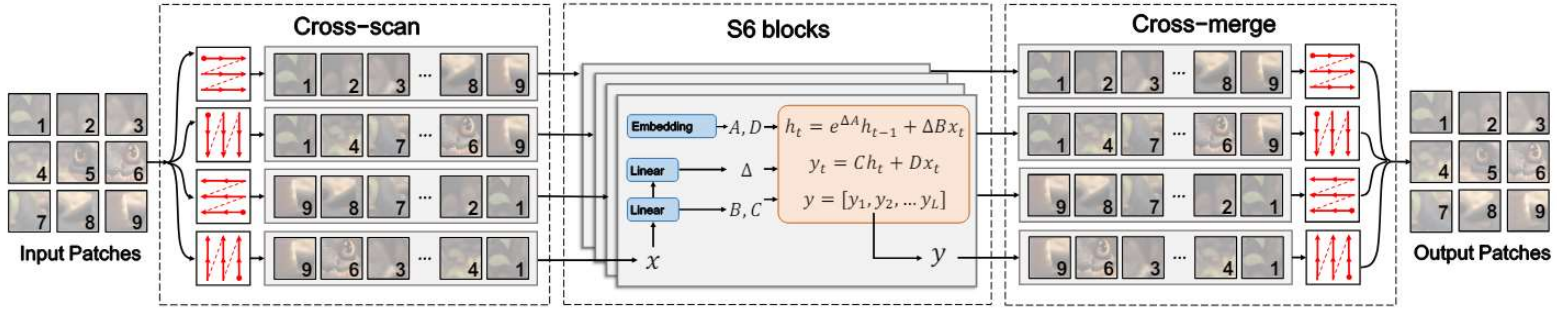
Vision Mamba

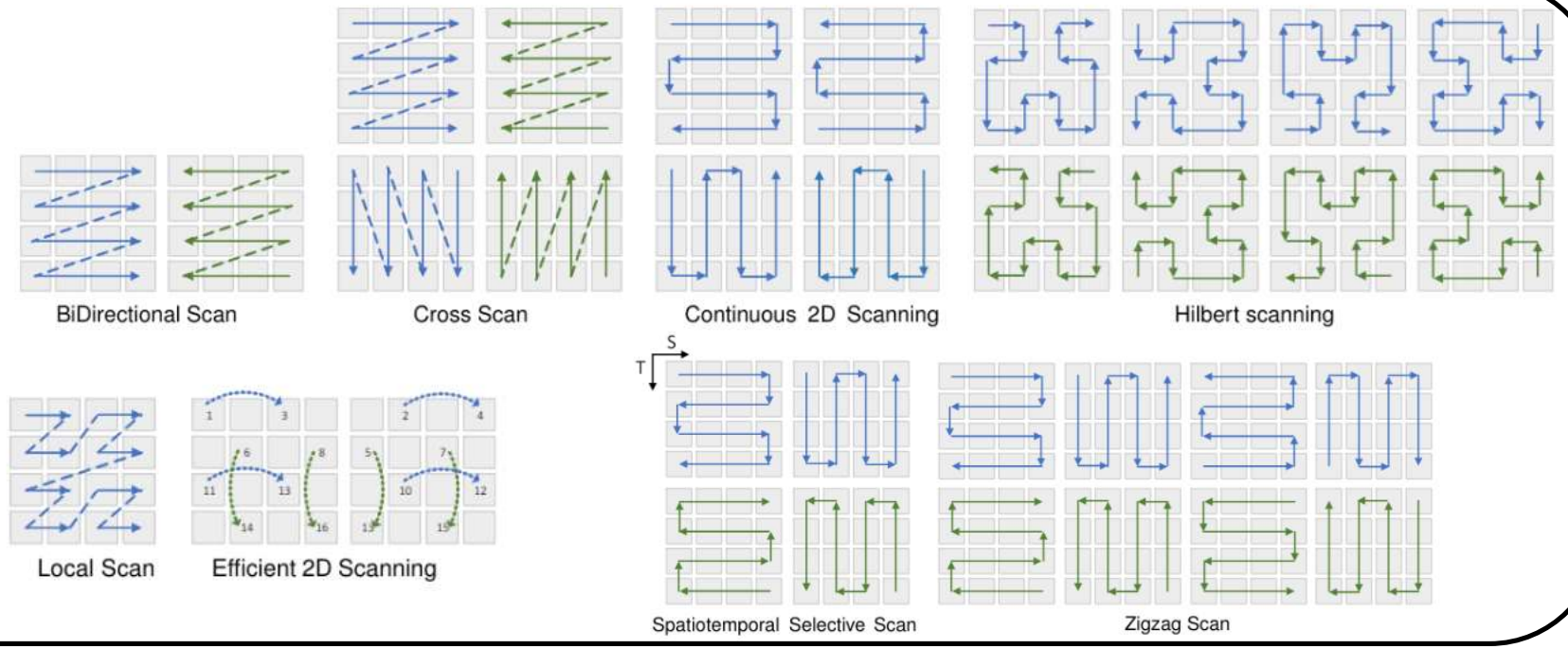


Zhu L, Liao B, Zhang Q, et al. Vision mamba: Efficient visual representation learning with bidirectional state space model[J]. arXiv preprint arXiv:2401.09417, 2024.

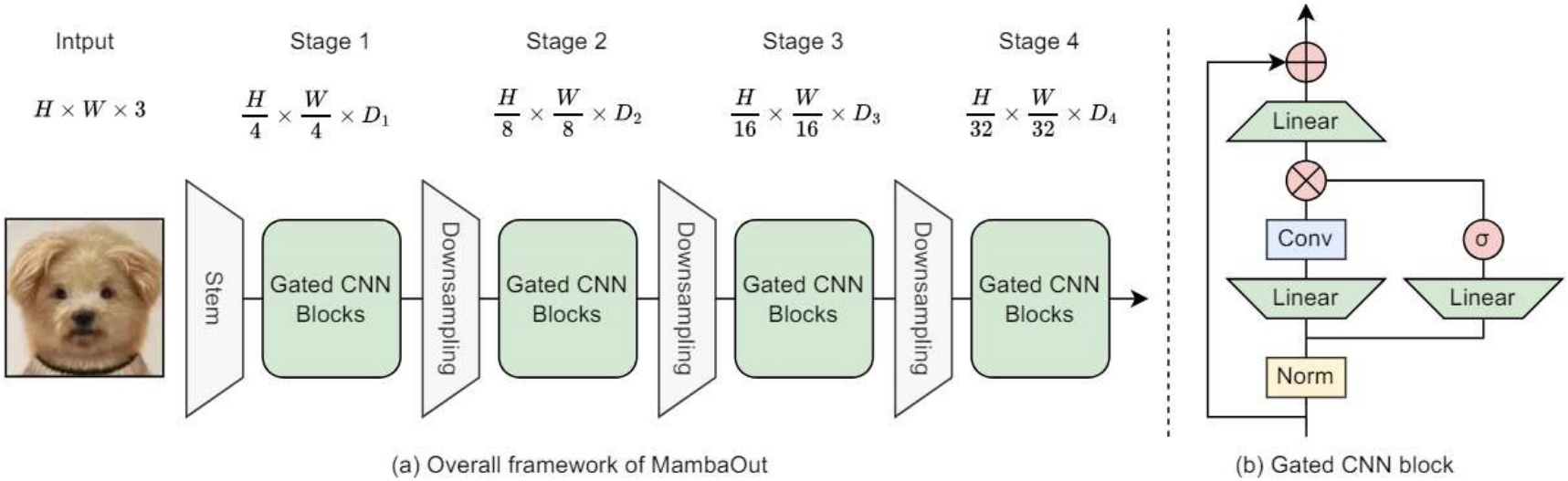
Mamba for CV

VMamba



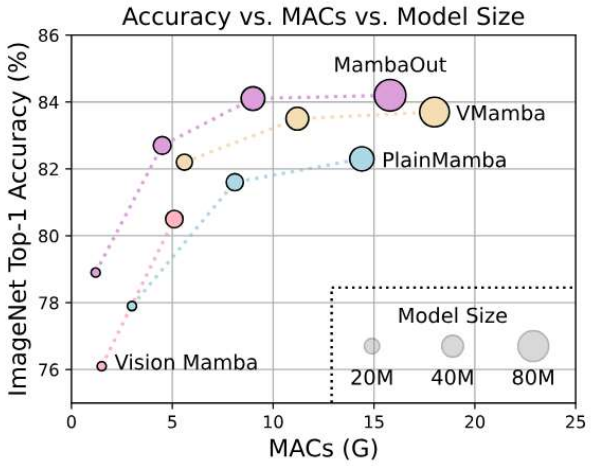


Mamba for CV



(a) Overall framework of MambaOut

(b) Gated CNN block



Yu W, Wang X. MambaOut: Do We Really Need Mamba for Vision?[J]. arXiv preprint arXiv:2405.07992, 2024.

Mamba for CV

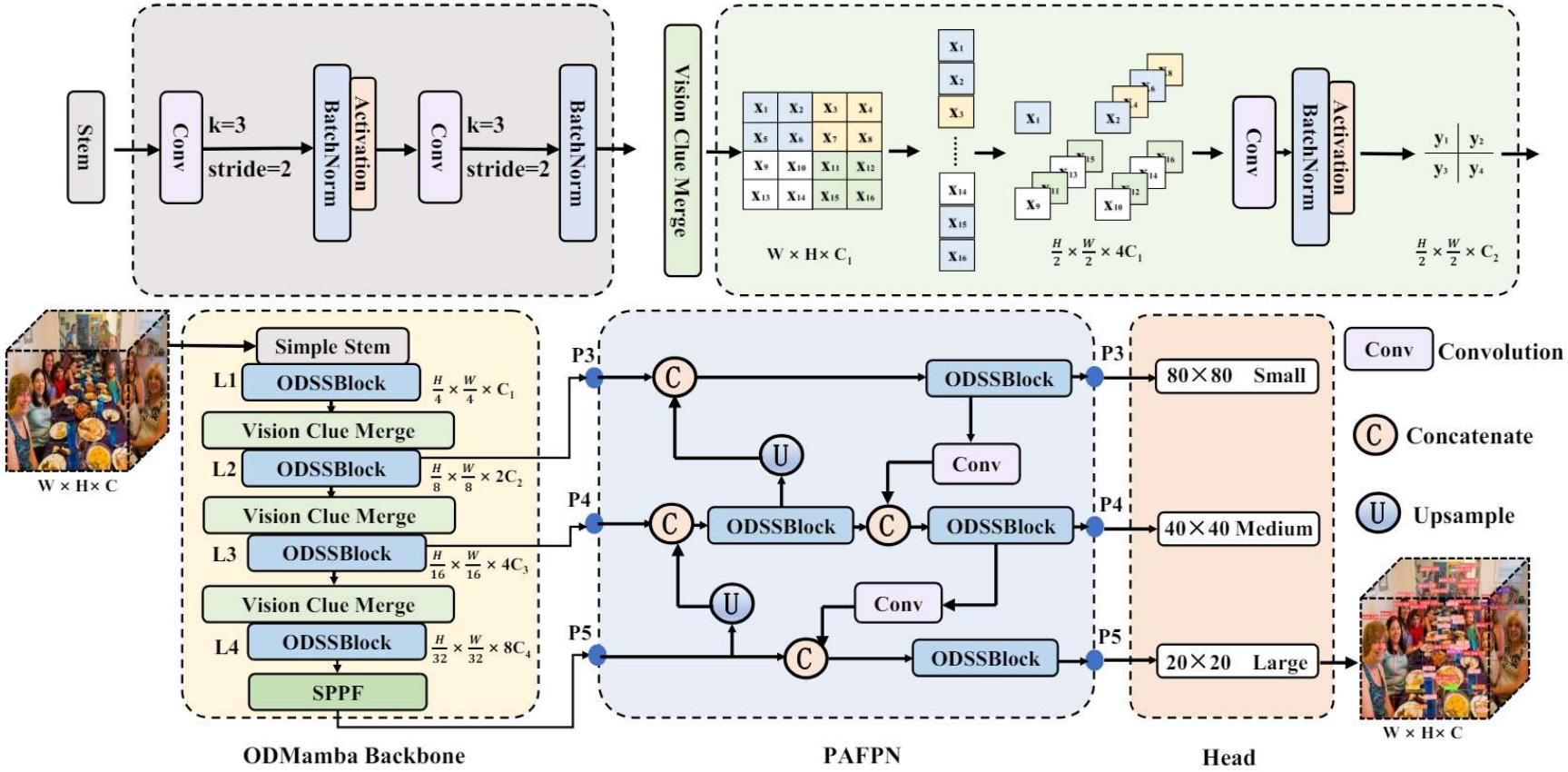


Figure 2: Illustration of Mamba YOLO architecture.

Wang Z, Li C, Xu H, et al. Mamba YOLO: SSMs-Based YOLO For Object Detection[J]. arXiv preprint arXiv:2406.05835, 2024.

Mamba for CV

Method	$AP^{val}(\%)$	$AP^{val}_{50}(\%)$	$AP^{val}_{75}(\%)$	$AP^{val}_{S}(\%)$	$AP^{val}_{M}(\%)$	$AP^{val}_{L}(\%)$	#param.	FLOPs
YOLOv5-N[46]	28.0	45.7	-	-	-	-	1.9 M	4.5 G
YOLOv5-S[46]	37.4	56.8	-	-	-	-	7.2 M	16.5 G
YOLOv5-M[46]	45.4	64.1	-	-	-	-	21.2 M	49.0 G
YOLOv5-L[46]	49.0	67.3	-	-	-	-	46.5 M	109.1 G
YOLOv6-3.0-N[27]	37.0	52.7	-	-	-	-	4.7 M	4.7 G
YOLOv6-3.0-S[27]	44.3	61.2	-	-	-	-	4.7 M	45.3 G
YOLOv6-3.0-M[27]	49.1	66.1	-	-	-	-	85.8 G	85.8 G
YOLOv6-3.0-L[27]	51.8	69.2	-	-	-	-	59.6 M	150.7 G
YOLOv7-Tiny[28]	37.4	55.2	37.3	15.7	38.0	53.4	6.2 M	13.7 G
YOLOv7[28]	51.2	69.7	55.9	31.8	55.5	65.0	36.9 M	104.7 G
YOLOv7-X[28]	52.9	71.7	51.4	36.9	57.7	68.6	71.3 M	189.9 G
YOLOv8-N[42]	37.3	52.6	40.6	18.8	41.0	53.5	3.2 M	8.7 G
YOLOv8-S[42]	44.9	61.8	48.6	26.0	49.9	61.0	11.2 M	28.6 G
YOLOv8-M[42]	50.2	67.3	54.8	32.3	55.9	66.5	25.9 M	78.9 G
YOLOv8-L[42]	52.9	69.8	57.7	35.5	58.5	69.8	43.7M	165.2 G
DAMO YOLO-T[30]	42.0	58.0	45.2	23.0	46.1	58.5	8.5 M	18.1 G
DAMO YOLO-S[30]	46.0	61.9	49.5	25.9	50.6	62.5	12.3 M	37.8 G
DAMO YOLO-M [30]	49.2	65.5	53.0	29.7	53.1	66.1	28.2 M	61.8 G
DAMO YOLO-L[30]	50.8	67.5	55.5	33.2	55.7	66.6	42.1 M	97.3 G
Gold-YOLO-N [29]	39.6	55.7	-	19.7	44.1	57.0	5.6 M	12.1 G
Gold-YOLO-S [29]	45.4	62.5	-	25.3	50.2	62.6	21.5 M	46.0 G
Gold-YOLO-M [29]	49.8	67.0	-	32.3	55.3	66.3	41.3 M	87.5 G
Gold-YOLO-L [29]	51.8	68.9	-	34.1	57.4	68.2	75.1 M	151.7 G
YOLO MS-XS [47]	43.4	60.4	47.6	23.7	48.3	60.3	4.5 M	17.4G
YOLO MS-S [47]	46.2	63.7	50.5	26.9	50.5	63.0	8.1 M	31.2 G
YOLO MS [47]	51.0	68.6	55.7	33.1	56.1	66.5	22.2 M	80.2 G
Mamba YOLO-T	45.4	62.3	49.1	25.2	50.4	62.9	6.1M	14.3G
Mamba YOLO-B	49.9	67.2	54.4	30.6	55.4	67.0	21.8M	49.7G
Mamba YOLO-L	52.1	69.8	56.5	34.1	57.3	68.1	57.6 M	156.2G

FPS?

Wang Z, Li C, Xu H, et al. Mamba YOLO: SSMs-Based YOLO For Object Detection[J]. arXiv preprint arXiv:2406.05835, 2024.