Advanced Transformer Architectures & Possible Alternatives

Team 6

Agenda

- State Space Model for New-Generation Network Alternative to Transformers: A Survey
- Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey
- FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness

UNIVERSITY of VIRGINIA | ENGINEERING
Department of Computer Science

← **Albert Gu**
234 posts

Follow

Albert Gu
@_albertgu

assistant prof @mldcmu. chief scientist @cartesia_ai. leading the ssm revolution.

Joined December 2018

Tri Dao

Tri Dao
Incoming Assistant Professor of Computer Science at Princeton University.
Chief Scientist at Together.AI.

Previously: PhD, Department of Computer Science
Stanford University
Advisors: Christopher Ré, Stefano Ermon
CV
Google Scholar

Highlights

- **Mamba: Linear-Time Sequence Modeling with Selective State Spaces**
  Albert Gu*, Tri Dao*.
  December 2023.
  [Paper] [Code]

- **FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness**
  Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, Christopher Ré.
  In *Hardware Aware Efficient Training Workshop at ICML*, 2022. **Best Paper Award**
  In *Sparsity in Neural Networks Workshop*, 2022. **Oral presentation**
  In *NeurIPS: Proceedings of the 35th Neural Information Processing Systems Conference*, December 2022.
  [Paper] [Code] [IEEE Spectrum and Forbes articles about our submission to the MLPerf 2.0 benchmark using FlashAttention]
  Usage: We've been very happy to see FlashAttention being widely adopted in such a short time after its release. This page contains a partial list of places where FlashAttention is being used.

University of Virginia | ENGINEERING
Department of Computer Science

State Space Model for New-Generation Network Alternative to Transformers: A Survey

Tongxuan Tian
nua3jz

Weifeng (Ellery) Yu
dag9wj

Content

- Motivation
- Formulation of SSM
- Structured State Space Model (S4)
- Mamba (S6)
- Variation of SSM

Motivation

- Self-attention mechanism enables transformer to learn **long-range feature representations** well.
- However, Transformer-based models require high-end GPU with **larger memory** for training and testing/deployment.

Motivation

- Self-attention mechanism enables transformer to learn **long-range feature representations** well.
- However, Transformer-based models require high-end GPU with **larger memory** for training and testing/deployment.

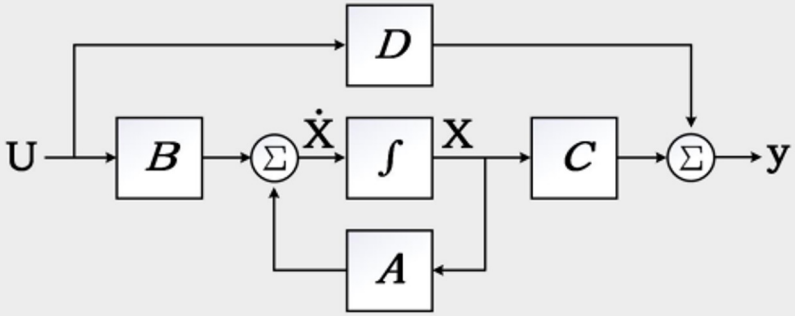
We need a model that not only requires **less computing cost** but also is still able to capture **long-range dependencies** while maintaining **high performance**.

State Space Model (SSM)

Formulation of SSM

$$\dot{\mathbf{X}}(t) = \mathbf{A}(t)\mathbf{X}(t) + \mathbf{B}(t)\mathbf{U}(t)$$
$$\mathbf{y}(t) = \mathbf{C}(t)\mathbf{X}(t) + \mathbf{D}(t)\mathbf{U}(t)$$

$$\dot{\mathbf{X}}(t) = \mathbf{A}(t)\mathbf{X}(t) + \mathbf{B}(t)\mathbf{U}(t)$$
$$\mathbf{y}(t) = \mathbf{C}(t)\mathbf{X}(t).$$



$$\dot{\mathbf{X}}(t) = \mathbf{A}(t)\mathbf{X}(t) + \mathbf{B}(t)\mathbf{U}(t)$$
$$\mathbf{y}(t) = \mathbf{C}(t)\mathbf{X}(t) + \mathbf{D}(t)\mathbf{U}(t)$$

A(t): state matrix
B(t): input matrix
C(t): output matrix
D(t): feed-forward matrix

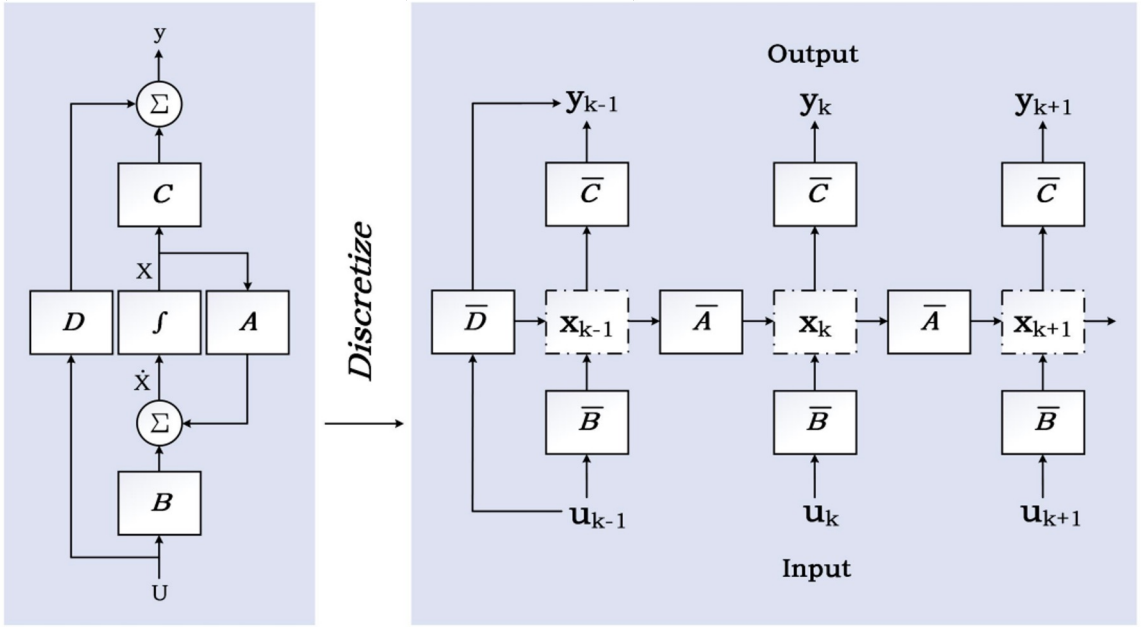
U(t): input vector
X(t): state vector
y(t): output vector

Linear State-Space Representation

Formulation of SSM

$$\dot{\mathbf{X}}(t) = \mathbf{A}(t)\mathbf{X}(t) + \mathbf{B}(t)\mathbf{U}(t)$$
$$\mathbf{y}(t) = \mathbf{C}(t)\mathbf{X}(t).$$

$$\mathbf{X}_t = \overline{\mathbf{A}}\mathbf{X}_{t-1} + \overline{\mathbf{B}}\mathbf{U}_t$$
$$\mathbf{y}_t = \mathbf{C}\mathbf{X}_t$$



Continuous-time

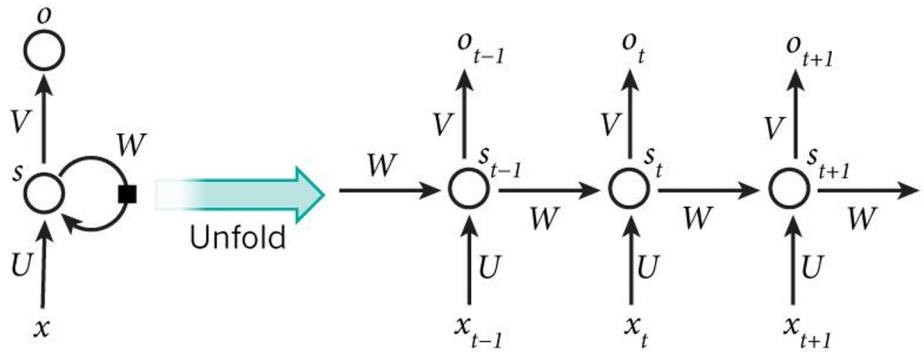
- ✓ *Continuous data*
- ✓ *Irregular sampling*

Recurrent

- ✓ *Unbounded context*
- ✓ *Efficient inference*

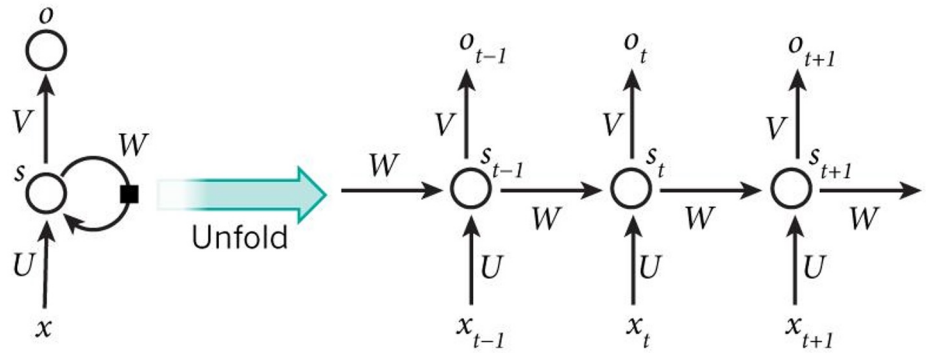
Formulation of SSM

$$\mathbf{h}_t = \overline{\mathbf{A}}\mathbf{h}_{t-1} + \overline{\mathbf{B}}\mathbf{x}_t \quad \Leftrightarrow \quad h_t = \mathrm{f}(Uh_{t-1} + Wx_t + b)$$
$$\mathbf{y}_t = \mathbf{C}\mathbf{h}_t. \qquad\qquad y_t = Vh_t$$



Formulation of SSM

$$\mathbf{h}_t = \overline{\mathbf{A}}\mathbf{h}_{t-1} + \overline{\mathbf{B}}\mathbf{x}_t \quad \Leftrightarrow \quad h_t = \mathrm{f}(Uh_{t-1} + Wx_t + b)$$
$$\mathbf{y}_t = \mathbf{C}\mathbf{h}_t. \qquad\qquad y_t = Vh_t$$



$$\mathbf{x}_t = \overline{\mathbf{A}}\mathbf{x}_{t-1} + \overline{\mathbf{B}}\mathbf{u}_t$$
$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t$$

$$\mathbf{y}_0 = \mathbf{C}\bar{\mathbf{A}}^0\bar{\mathbf{B}}\mathbf{x}_0$$
$$\mathbf{y}_1 = \mathbf{C}\bar{\mathbf{A}}^1\bar{\mathbf{B}}\mathbf{x}_0 + \mathbf{C}\bar{\mathbf{A}}^0\bar{\mathbf{B}}\mathbf{x}_1$$
$$\mathbf{y}_2 = \mathbf{C}\bar{\mathbf{A}}^2\bar{\mathbf{B}}\mathbf{x}_0 + \mathbf{C}\bar{\mathbf{A}}^1\bar{\mathbf{B}}\mathbf{x}_1 + \mathbf{C}\bar{\mathbf{A}}^0\bar{\mathbf{B}}\mathbf{x}_2.$$

Convolutional Form

$$\overline{\mathbf{K}} = (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}\mathbf{B}}, \ldots, \mathbf{C}\overline{\mathbf{A}}^k\overline{\mathbf{B}}, \ldots)$$
$$\mathbf{y} = \mathbf{x} * \overline{\mathbf{K}}.$$

SSM

- Similar to RNNs, SSM also suffers from the **vanishing/exploding gradients problem** when modeling **longer** sequences.

HiPPO

- Combines the concepts of Recurrent Memory and Optimal Polynomial Projections, which can significantly improve the performance of recursive memory.

$$A_{nk} = \begin{cases} (2n+1)^{1/2}(2k+1)1/2 & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}$$

Vanilla SSM + HiPPO ⇨ Structured State Space Model (S4)

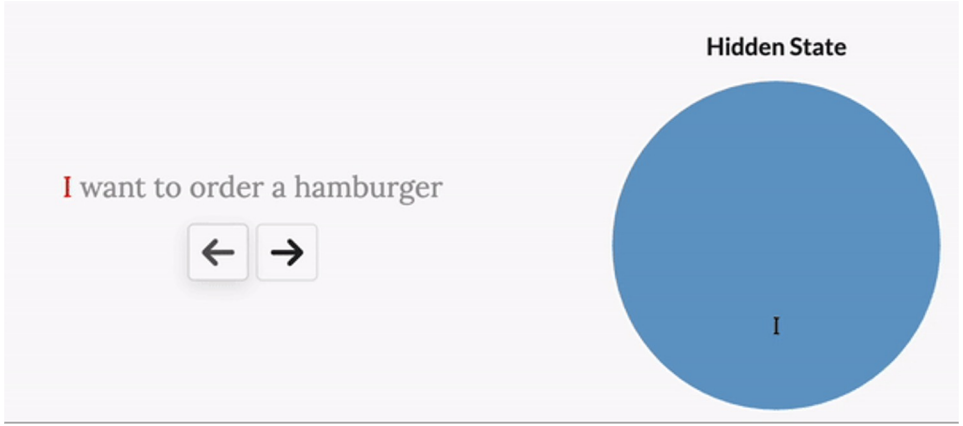
From S4 to Mamba (S6)

$$\mathbf{h}_t = \overline{\mathbf{A}}\mathbf{h}_{t-1} + \overline{\mathbf{B}}\mathbf{x}_t$$
$$\mathbf{y}_t = \mathbf{C}\mathbf{h}_t.$$

- S4 does not have **selectivity**
- Those discrete parameters are constant.

- Mamba makes these parameters vary based on the input.

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



Parallization

- In S4, we could **precompute** this kernel, **save** it, and **multiply** it with the input x.

$$\overline{\mathbf{K}} = (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}\mathbf{B}}, ..., \mathbf{C}\overline{\mathbf{A}}^{k}\overline{\mathbf{B}}, ...)$$

$$\mathbf{y} = \mathbf{x} * \overline{\mathbf{K}}.$$

Parallization

- In S4, we could **precompute** this kernel, **save** it, and **multiply** it with the input x.

$$\overline{\mathbf{K}} = (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}\mathbf{B}}, ..., \mathbf{C}\overline{\mathbf{A}}^{k}\overline{\mathbf{B}}, ...)$$

$$\mathbf{y} = \mathbf{x} * \overline{\mathbf{K}}.$$

- In Mamba, these matrices change depending on the input.
- If we want selectivity, we'll need to train with **RNN mode**.

~~$\mathbf{y} = \mathbf{x} * \overline{\mathbf{K}}.$~~

Parallel Scan

Whether an operation can be done in parallel depends on **associative property**.

$$m = (\bar{A}_0, \bar{B}_0 x_0) \quad n = (\bar{A}_1, \bar{B}_1 x_1)$$

$$\begin{aligned} m \oplus n &= (\bar{A}_0, \bar{B}_0 x_0) \oplus (\bar{A}_1, \bar{B}_1 x_1) \\ &= (\bar{A}_0 \bar{A}_1, \bar{A}_1 \bar{B}_0 x_0 + \bar{B}_1 x_1) \end{aligned}$$

$$k = (\bar{A}_3, \bar{B}_3 x_3)$$

$$(m \oplus n) \oplus k = m \oplus (n \oplus k)$$

Parallel Scan

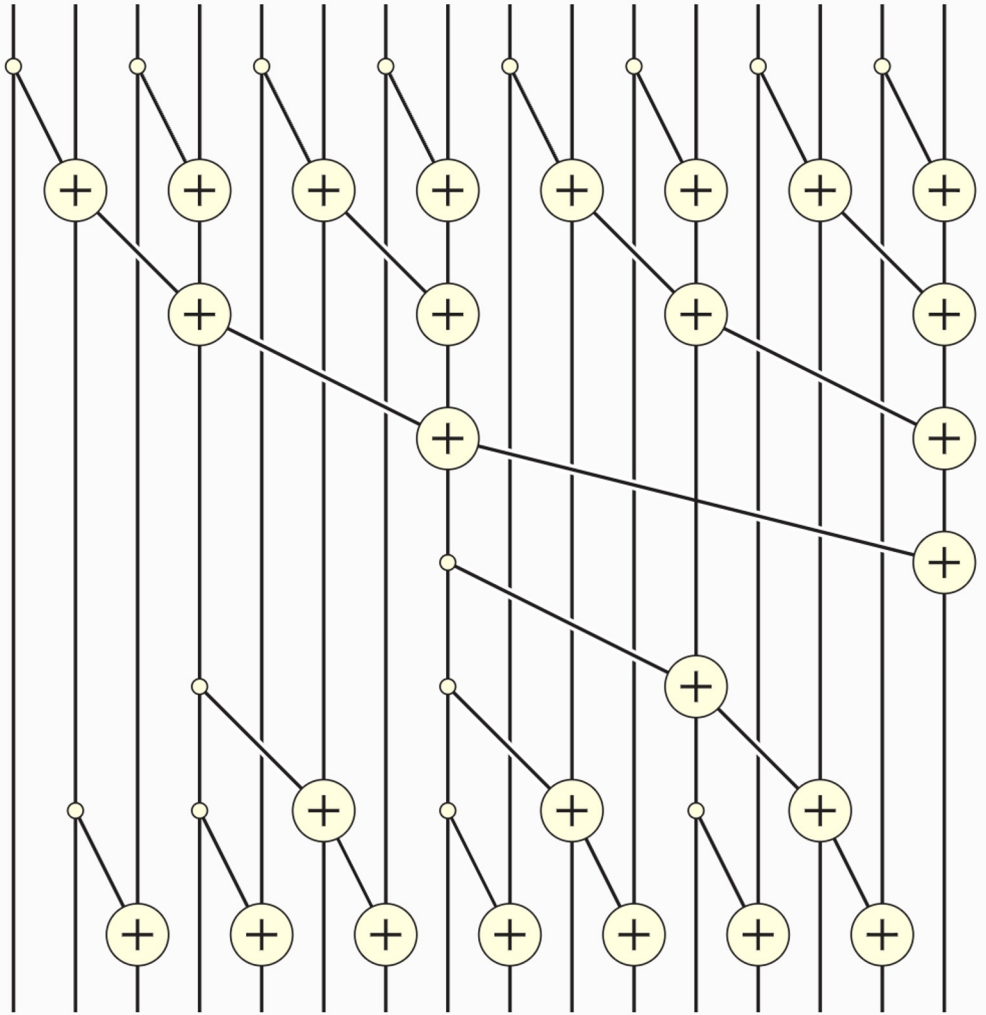
Whether an operation can be done in parallel depends on **associative property**.

$$m = (\bar{A}_0, \bar{B}_0 x_0) \quad n = (\bar{A}_1, \bar{B}_1 x_1)$$

$$\begin{aligned} m \oplus n &= (\bar{A}_0, \bar{B}_0 x_0) \oplus (\bar{A}_1, \bar{B}_1 x_1) \\ &= (\bar{A}_0 \bar{A}_1, \bar{A}_1 \bar{B}_0 x_0 + \bar{B}_1 x_1) \end{aligned}$$

$$k = (\bar{A}_3, \bar{B}_3 x_3)$$

$$(m \oplus n) \oplus k = m \oplus (n \oplus k)$$



Variations in Natural Language Processing

- Language Modeling
- Voice
- Clinical note
- Translation

8	**GSS** [41]	arXiv22	NLP	GSS-192M GSS-L-352M GSS-Hybrid-L-373M	SSM-former	Language modeling	Perplexity 10.52(arXiv)(↓)	5.6(steps/sec)	URL
9	**Spiking-S4** [42]	arXiv23	NLP	0.53M	S4	Deep Noise Suppression	SISNR 14.58(DNS 2023)(↑) WB-PESQ 3.39 (Voice-Bank+Demand)(↑)	FLOPs:$1.5 * 10^9$	-
10	**DPMamba** [43]	arXiv24	NLP	DPMamba-XS 2.3M DPMamba-S 8.1M DPMamba-M 15.9M DPMamba-L 59.8M	Mamba	Speech Separation	SISNR 24.4(WSJ0-2mix)(↑)	-	-
11	**SPMamba** [44]	arXiv24	NLP	6.14M	Mamba	Speech Separation	SISNR 15.20 (SPMamba self-built)	Macs 78.69	URL
12	**Grazzi et al.** [45]	arXiv24	NLP	–	Mamba	Algorithmic Knowledge Linguistic Translation	Acc 0.93(Next letter) Acc 0.82(Location continent) Acc 0.90(Plural singular) Acc 0.78(En es)	–	-
13	**S4++** [46]	-	NLP	ALM S4++: 49.85M BLM S4++: 134.11M	S4	Language Modeling Long-Range Dependency Modeling	PPL 25.31(WikiText-103)(↓) Avg 76.61(CLUE)(↑) Avg 78.31(LRA)(↑)	–	-
14	**SPADE** [47]	-	NLP	SPADEbase++: 290M	S4+Transformer	Language Modeling Long-Range Dependency Modeling Language Generation	PPL 18.5(WikiText-103)(↓) Avg 87.40(LRA)(↑) Avg 86.8(CLUE)(↑) ROUGE-2 21.65(arXiv)(↑)	sequence length 6k Memory: 27G	URL
15	**DenseMamba** [48]	arXiv24	NLP	DenseMamba-360M DenseMamba-1.3B	Mamba	Common-Sense Reasoning Question-Answering	Zero-shot Avg 54.51 (ARC_E,ARC_C) four-shot Avg 55.05 (ARC_E,ARC_C)	-	URL
16	**ClinicalMamba** [49]	arXiv24	NLP	ClinicalMamba-130M ClinicalMamba-2.8B	Mamba	Cohort Selection ICD Coding	Prec 88.6(n2c2 challenge in 2018) Prec 75.28(Code-rare) Prec 75.53(Code-common)	-	URL
17	**MDLT** [50]	arXiv24	NLP	-	Mamba	Translation	PhantomDance MDLT-T AJE:0.87 ± 0.02 FID: 0.39 ± 0.02 MDLT-M AJE:0.73 ± 0.01 FID:0.82 ± 0.01	-	URL

Language Modeling

Gate State Space:

- Can be used on long sequence modeling.
- Reducing number of participants.
- 2-3 times faster than Diagonal State Space.

S4+++:

- State Memory Relay.
- Integrate complex dependency bias via an interactive cross-validation mechanism.

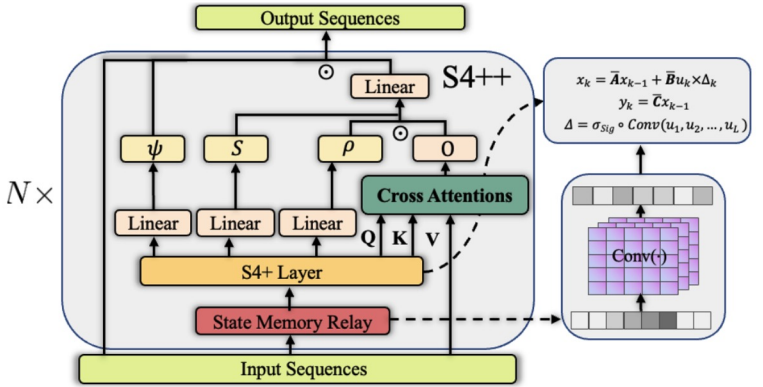


Figure 5: Illustration of the S4++ layer.

Voice Task

DP-Mamba

- **Bidirectional Dependency Modeling:** Simultaneously models both **short-term** and **long-term** forward and backward dependencies of speech signals.
- **Selective State Space:** Enhances model capability through a **selectively utilized state space**.
- **Performance**: Achieves comparable results to the dual-path Transformer model Sepformer.

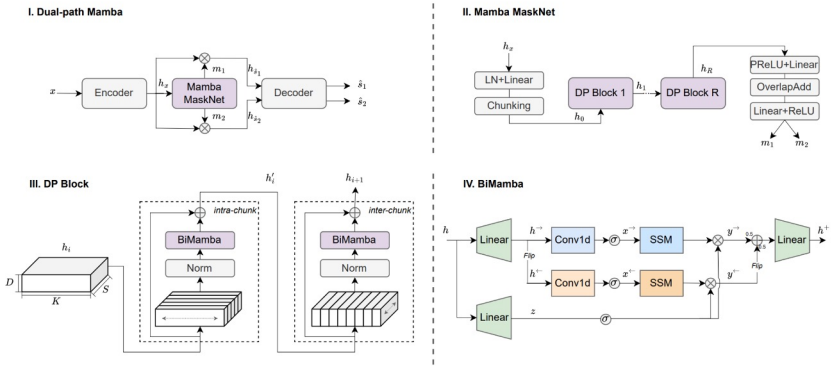


Fig. 1. A top-down view of DPMamba from I to IV.

SP-Mamba:

- Utilizes **TF-GridNet**.
- Replaces the Transformer module with a **bidirectional Mamba module**.
- **Result:** Captures a wider range of language information, leading to broader comprehension.

Variations in Clinical Notes

Mamba is capable of modeling very **long sequences of clinical notes**, up to 16,000 characters.

ClinicalMamba

- Uses the MIMIC-III dataset for pre-training the Mamba model.
- Better than **Mamba** and **clinical Llama** models in modeling clinical language, especially on longer text lengths.

Variations in Computer Vision

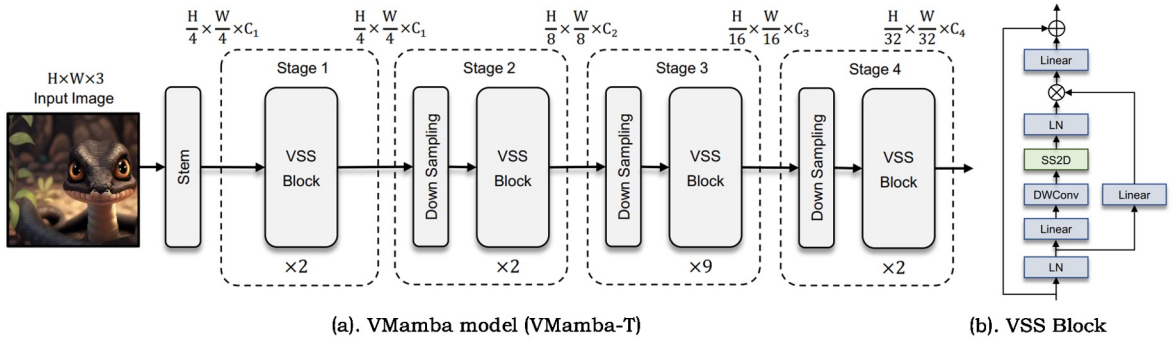
Vast grown

In the paper, the researcher list 148 SSM-based models, most of which are in the field of Computer Vision.

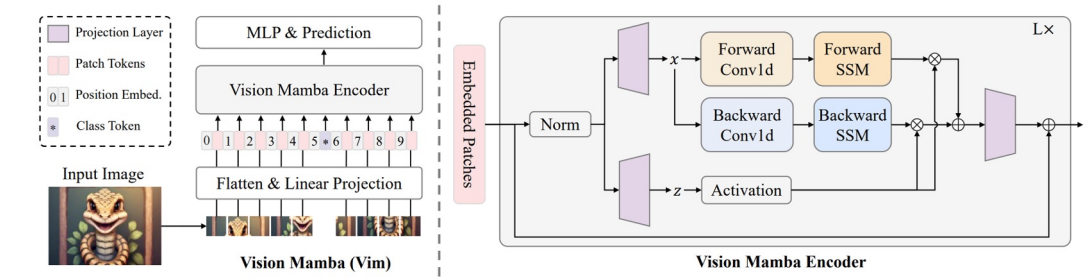
Variations in Computer Vision

VMamba

Vision Mambda(vim)



(a). VMamba model (VMamba-T) (b). VSS Block



(c). Vision Mamba (Vim) and Detailed Atchitecture

Vision Mamba (Vim)

transform the 2-D image $\mathbf{t} \in \mathbb{R}^{H \times W \times C}$ into the flattened 2-D patches $\mathbf{x_p} \in \mathbb{R}^{J \times (P^2 \cdot C)}$,

where **P** is size of patches. Then project the 2-Dpatches to the vector with size D

$$\mathbf{E}_{pos} \in \mathbb{R}^{(J+1) \times D}$$

$$\mathbf{W} \in \mathbb{R}^{(P^2 \cdot C) \times D}$$

$$\mathbf{T}_0 = [\mathbf{t}_{cls}; \mathbf{t}_p^1\mathbf{W}; \mathbf{t}_p^2\mathbf{W}; \cdots; \mathbf{t}_p^J\mathbf{W}] + \mathbf{E}_{pos}$$

$$\mathbf{T}_l = \mathbf{Vim}(\mathbf{T}_{l-1}) + \mathbf{T}_{l-1},$$

$$\mathbf{f} = \mathbf{Norm}(\mathbf{T}_L^0),$$

$$\hat{p} = \mathbf{MLP}(\mathbf{f}),$$

Vision Mamba (Vim)

$$\mathbf{T}_l = \mathbf{Vim}(\mathbf{T_{1-1}}) + \mathbf{T_{1-1}},$$
$$\mathbf{f} = \mathbf{Norm}(\mathbf{T}_\mathbf{L}^0),$$
$$\hat{p} = \mathbf{MLP}(\mathbf{f}),$$

Algorithm 1 Vim Block Process

Require: token sequence $\mathbf{T}_{l-1}$: (B, M, D)
Ensure: token sequence $\mathbf{T}_l$: (B, M, D)

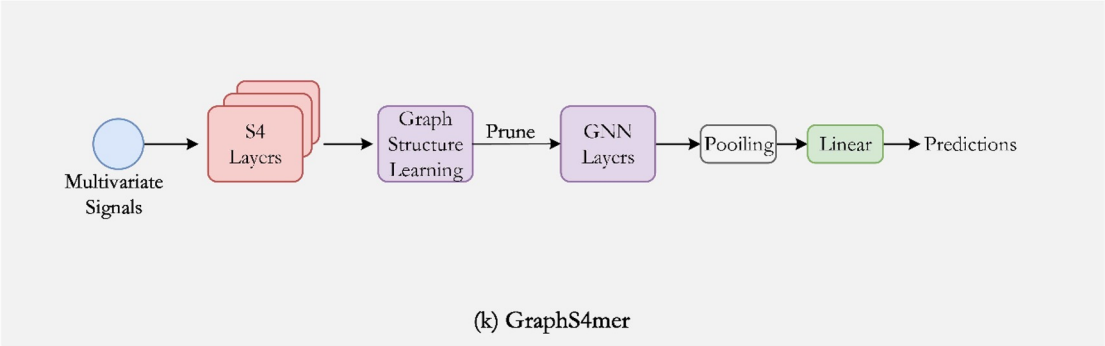
/* normalize the input sequence $\mathbf{T}'_{l-1}$ */
$\mathbf{T}'_{l-1}$: (B, M, D) $\leftarrow \mathbf{Norm}(\mathbf{T}_{l-1})$
$\mathbf{x}$: (B, M, E) $\leftarrow \mathbf{Linear}^{\mathbf{x}}(\mathbf{T}'_{l-1})$
$\mathbf{z}$: (B, M, E) $\leftarrow \mathbf{Linear}^{\mathbf{z}}(\mathbf{T}'_{l-1})$
/* process with different direction */
for o in {forward, backward} **do**
  $\mathbf{x}'_o$: (B, M, E) $\leftarrow \mathbf{SiLU}(\mathbf{Conv1d}_o(\mathbf{x}))$
  $\mathbf{B}_o$: (B, M, N) $\leftarrow \mathbf{Linear}_o^{\mathbf{B}}(\mathbf{x}'_o)$
  $\mathbf{C}_o$: (B, M, N) $\leftarrow \mathbf{Linear}_o^{\mathbf{C}}(\mathbf{x}'_o)$
  /* softplus ensures positive $\mathbf{\Delta}_o$ */
  $\mathbf{\Delta}_o$: (B, M, E) $\leftarrow \log(1 + \exp(\mathbf{Linear}_o^{\mathbf{\Delta}}(\mathbf{x}'_o) + \mathbf{Parameter}_o^{\mathbf{\Delta}}))$
  /* shape of $\mathbf{Parameter}_o^{\mathbf{A}}$ is (E, N) */
  $\overline{\mathbf{A}}_o$: (B, M, E, N) $\leftarrow \mathbf{\Delta}_o \bigotimes \mathbf{Parameter}_o^{\mathbf{A}}$
  $\overline{\mathbf{B}}_o$: (B, M, E, N) $\leftarrow \mathbf{\Delta}_o \bigotimes \mathbf{B}_o$
  $\mathbf{y}_o$: (B, M, E) $\leftarrow \mathbf{SSM}(\overline{\mathbf{A}}_o, \overline{\mathbf{B}}_o, \mathbf{C}_o)(\mathbf{x}'_o)$
end for
/* get gated $\mathbf{y}_o$ */
$\mathbf{y}'_{forward}$: (B, M, E) $\leftarrow \mathbf{y}_{forward} \odot \mathbf{SiLU}(\mathbf{z})$
$\mathbf{y}'_{backward}$: (B, M, E) $\leftarrow \mathbf{y}_{backward} \odot \mathbf{SiLU}(\mathbf{z})$
/* residual connection */
$\mathbf{T}_l$: (B, M, D) $\leftarrow \mathbf{Linear}^{\mathbf{T}}(\mathbf{y}'_{forward} + \mathbf{y}'_{backward}) + \mathbf{T}_{l-1}$
Return: $\mathbf{T}_l$

Variations in Computer Vision

- **Classification task**
- **Detection task:** MiM-ISTD
- **Segmentation task**
    a. Medical image segmentation: VM-UNet
- **Medical tasks**
    a. Registration task: MambaMorph
- **Restoration task:** MambdaIR
- **Generation task:** ZigMa
- **Video understanding:**ViS4mer, Video Mamba

Variations in Graph

GraphS4mer uses the S4 architecture to capture long-range dependencies and includes a dynamic graph structure learning layer for spatial correlations.



(k) GraphS4mer

GMN is based on selective State Space Models, tackling the limitations of traditional GNNs in capturing long-range dependencies and computational efficiency.

Viriations in Multi-modaliday and Multi-media

- **S4ND Model:**
  - Extends State Space Models to **multidimensional** signals.
  - Effective in large-scale visual data modeling across 1D, 2D, and 3D dimensions.
  - Proven applications in image and video classification.
- **VL-Mamba:**
  - First implementation of the state-space model Mamba in multimodal tasks.
  - Aims to address high computational costs in Transformer architectures.
- **CMViM:**
  - Focuses on multimodal learning for 3D high-resolution medical images, specifically Alzheimer's disease.
  - Utilizes the MAE framework, replacing the ViT module with a simpler **Vim** module to reduce computational complexity from quadratic to linear.
  - Enhances modeling capabilities through **intra-modality** and i**nter-modality** contrastive learning, improving feature discrimination and aligning representations across different modalities.

Variation for Time Serires

TimeMachine

Purpose: Addresses challenges in long-term time-series forecasting (LTSF).
Key Challenges:

- Capturing long-term dependency relationships.
- Overcoming poor linear scalability in time-series data.

Innovative Solution:

- Uses multiple Mamba modules integrated into a singular architecture to enhance dependency capture and improve channel mixing.
- Provides selective prediction capabilities for both global and local contexts across various scales.

Results: Demonstrated significant improvements in accuracy and scalability in experimental validations.

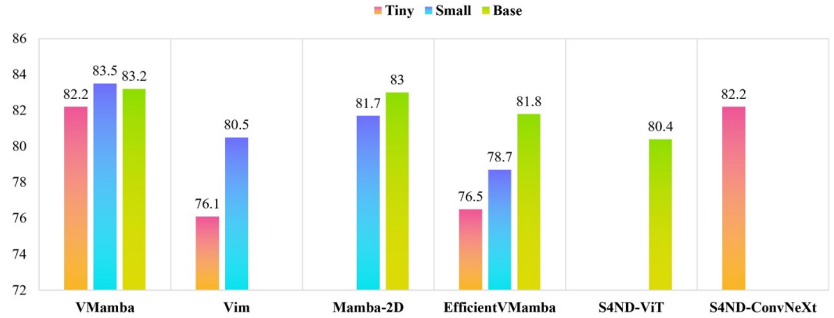
Experiment & Results

To evaluate the performance practically, the experiments involved five downstream tasks:

- **single-/multi-label classification**
- **visual object tracking**
- pixel-level segmen- tation
- image-to-text generation
- person/vehicle re- identification

Single-/Multi-label Classification

Single-label classification: calculate accuracy on ImageNet-1K.



Multilabel classification：Evaluate performance on Pedestrain Attribute Recognition task.

TABLE 10
Comparison with state-of-the-art methods on PETA and PA100K datasets. The first and second are shown in red and blue, respectively. "-" means this indicator is not available. VTB* indicates that VTB uses CLIP's feature extractor.

Methods	Backbone	PETA					PA100K				
		mA	Acc	Prec	Recall	F1	mA	Acc	Prec	Recall	F1
JLAC (AAAI 2020) [195]	ResNet50	86.96	80.38	87.81	87.09	87.50	82.31	79.47	87.45	87.77	87.61
SCRL (TCSVT 2020) [196]	ResNet50	87.2	-	89.20	87.5	88.3	80.6	-	88.7	84.9	82.1
SSCsoft (ICCV 2021) [197]	ResNet50	86.52	78.95	86.02	87.12	86.99	81.87	78.89	85.98	89.10	86.87
IAA-Caps (PR 2022) [198]	OSNet	85.27	78.04	86.08	85.80	85.64	81.94	80.31	88.36	88.01	87.80
MCFL (NCA 2022) [199]	ResNet-50	86.83	78.89	84.57	88.84	86.65	81.53	77.80	85.11	88.20	86.62
DRFormer (NC 2022) [200]	ViT-B/16	89.96	81.30	85.68	91.08	88.30	82.47	80.27	87.60	88.49	88.04
VAC-Combine (IJCV 2022) [201]	ResNet50	-	-	-	-	-	82.19	80.66	88.72	88.10	88.41
DAFL (AAAI 2022) [202]	ResNet50	87.07	78.88	85.78	87.03	86.40	83.54	80.13	87.01	89.19	88.09
CGCN (TMM 2022) [203]	ResNet	87.08	79.30	83.97	89.38	86.59	-	-	-	-	-
CAS-SAL-FR (IJCV 2022) [204]	ResNet50	86.40	79.93	87.03	87.33	87.18	82.86	79.64	86.81	87.79	85.18
PromptPAR (arXiv24) [205]	ViT-L/14	88.76	82.84	89.04	89.74	89.18	87.47	83.78	89.27	91.70	90.15
SequencePAR (arXiv24) [206]	ViT-L/14	-	84.92	90.44	90.73	90.46	-	83.94	90.38	90.23	90.10
VTB (TCSVT 2022) [207]	ViT-B/16	85.31	79.60	86.76	87.17	86.71	83.72	80.89	87.88	89.30	88.21
VTB* (TCSVT 2022) [207]	ViT-L/14	86.34	79.59	86.66	87.82	86.97	85.30	81.76	87.87	90.67	88.86
VTB (TCSVT 2022) [207]	ViT-S	82.51	77.23	85.75	84.95	85.01	78.76	77.61	87.41	85.35	85.94
Vim-PAR	Vim-S	81.08	73.75	80.91	84.96	82.52	80.41	78.03	85.39	88.37	86.39

Visual Object Tracking

Compare Mamba with Transformer and CNN based backbon for the tracking tasks.

TABLE 11
Comparison between different trackers on the EventVOT dataset.

Trackers	Source	Backbone	SR	PR	NPR	Params(M)	FPS
TrDiMP	CVPR21	ResNet50	39.9	34.8	48.7	26.3	26
ToMP50	CVPR22		37.6	32.8	47.4	26.1	25
DiMP50	ICCV19		52.6	51.1	67.2	26.1	43
PrDiMP	CVPR20		55.5	57.2	70.4	26.1	30
KYS	ECCV20		38.7	37.3	49.8	–	20
ATOM	CVPR19		44.4	44.0	57.5	8.4	30
HDETrack	CVPR24	ViT	57.8	62.2	73.5	92.1	105
AiATrack	ECCV22		57.4	59.7	72.8	15.8	38
STARK	ICCV21		44.5	39.6	55.7	28.1	42
TransT	CVPR21		54.3	56.5	68.8	18.5	50
MixFormer	CVPR22		49.9	49.6	63.0	35.6	25
SimTrack	ECCV22		55.4	57.5	69.9	57.8	40
OSTrack	ECCV22	ViT-B	55.4	60.4	71.1	92.1	105
		ViT-S	52.0	53.2	66.8	54.3	109
		Vim-S	55.6	59.1	70.4	4.1	41

Challenges

- Current SSMs model still performs inferior to the main- stream of Transformer networks.
- Multi-modal learning using SSMs architecture.
- How to design new SSMs- based backbones for cost-sensitive multi-modal learning is an important research topic.
- How to use the latest SSM model to empower the existing deep neural network model?

Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey

Nanjing University, University of London, Baidu Inc.

Feng Guo(grj4jc), Yanxi Liu(kww7url)

Agenda

1. Introduction
2. Overview
3. Efficient Attention
4. Long-Term Memory
5. Extrapolative PEs
6. Context Processing
7. Miscellaneous
8. Evaluation Necessity & Optimization Toolkit
9. Discussion
10. Conclusion

Introduction to Long-Context LLMs

- Great Success for Transformer-based LLM Models (chatGPT, Bert, Claude..)
  - Indicates a potential path towards AGI
  - Revolutionizing Application: Document summarization, Computer vision, …
  - Essential for advanced applications
    - like detailed text analysis and interactive AI systems
- Success due to well-designed **Attention Mechanism,** but …

Challenges and Research Directions in Long-Context LLMs

- **Challenges in Current Transformer Models**
  - **Complexities**: High computational needs with quadratic time and space complexities during training and inference
  - **Performance Degradation**: Lack of robustness in mechanism leads to **performance degradation** with long sequences
- **Research Directions**
  - **Efficiency Improvements**: Attention mechanism, memory mechanisms
  - **Handling Long Contexts**: Effective length generalization, context pre/post processing

Contributions of this Survey

- **Holistic Taxonomy**:Detailed breakdown of Transformer architecture enhancements.
- **Evaluations and Toolkits**: Analysis of datasets, metrics, libraries, frameworks for optimizing LLM efficiency.
- **Future Directions**: Identifying key challenges and potential solutions for advancing long-context comprehension in LLMs.

Section 2: Overview

- Preliminaries of LLM
- LLMs Architecture limitations
- Road Map of Enhance Long-Context Capabilities in LLMs

Preliminaries of Neural Language Modeling

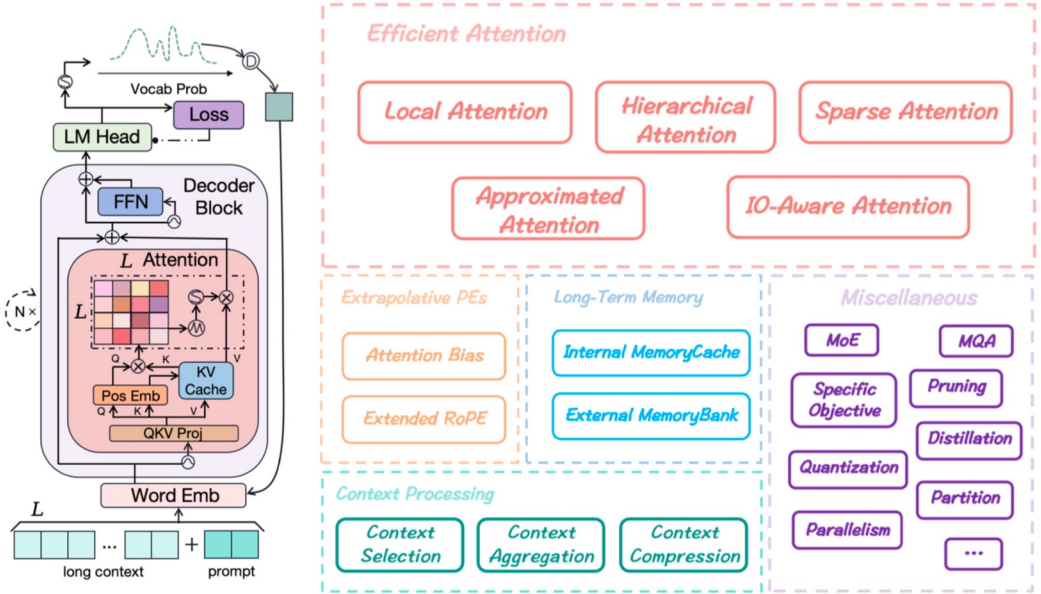
- **Modeling Stages**
  - **Preprocessing**: Tokenization of raw text into subwords or tokens
  - **Pretraining**: Learning semantic patterns and linguistic structures on large corpora
  - **Fine-tuning**: Adapting the pre-trained model to task-specific data for downstream applications
  - **Inference**: Auto regressively generating text based on learned probabilities

- **Key-Value Cache in LLMs**
  - **Functionality**: Stores key-value pairs for attention, extending sequences during generation
  - **Limitation**: Linearly growing memory occupation with generated tokens, prompting long-term memory enhancements

Limitations of Transformer Architecture in Handling Long Contexts

- **Attention Complexity**
  - Computational Complexity: In scenarios where **sequence length** L far exceeds **dimension** d,
    - The complexity becomes quadratic
    - Time Complexity: $O(L$^2*d) Space Complexity: $O(L$^2)
- **In-context Memory Limitations**
  - **Statelessness of Transformers**: **Lacks a mechanism to retain state** between calls, relying only on the KV cache
  - **Impact on Applications**: This design limits effectiveness in applications requiring long-term memory(chatbots)
- **Max-Length Constraint**
  - **Training and Inference**: **Engineers set a maximum sequence length** $Lmax$ to prevent memory overflow
    - As a hyper-parameter, typically between 1K, 2K 4K tokens
  - **Performance degradation**: observed when handling inputs longer than $Lmax$ resulting in **implausible outputs**

Roadmap of Enhancements for Long-Context Capabilities in LLMs



- **Efficient Attention** (Sec. 3)
- **Long-Term Memory** (Sec. 4)
- **Extrapolative Positional Encodings**(Sec. 5)
- **Context Processing** (Sec. 6)
- **Miscellaneous** (Sec. 7)

- **Reduce attention complexity**
- **designing efficient memory mechanisms**
- **Enhancing long context capabilities**

Section 3: Efficient Attention Mechanisms

- **Goal**: Addressing the **computational bottleneck of attention mechanisms** in Transformers
- **Impact**: **Expanding the context length boundary** for LLMs during both pre training and inference phases
- **Category**
  a. **Local Attention**
  b. **Hierarchical Attention**
  c. **Sparse Attention**
  d. **Approximated Attention**
  e. **IO-Aware Attention**

Local Attention

- **Redefining Attention Mechanisms**
  - **Traditional Global Attention**: Each token attends to all others, leading to $O(L^2d)$ complexities
  - **Local Attention**: Focuses on neighboring tokens, reducing time and space complexities
- Approaches
  - **Block-wise Attention**
    - Divides input into non-overlapping blocks, each attending within itself(e.g. BlockBERT)
  - **Sliding Window Attention**
    - Each token attends within a fixed-size window, inspired by CNN techniques(e.g. Longformer)
  - **Global-Local Hybrid Attention**
    - Combines local attention **with global tokens** for broader context (e.g. LongLM)
  - **LSH Attention**
    - Utilizes **locality-sensitive hashing** for efficient neighbor token selection

Local Attention

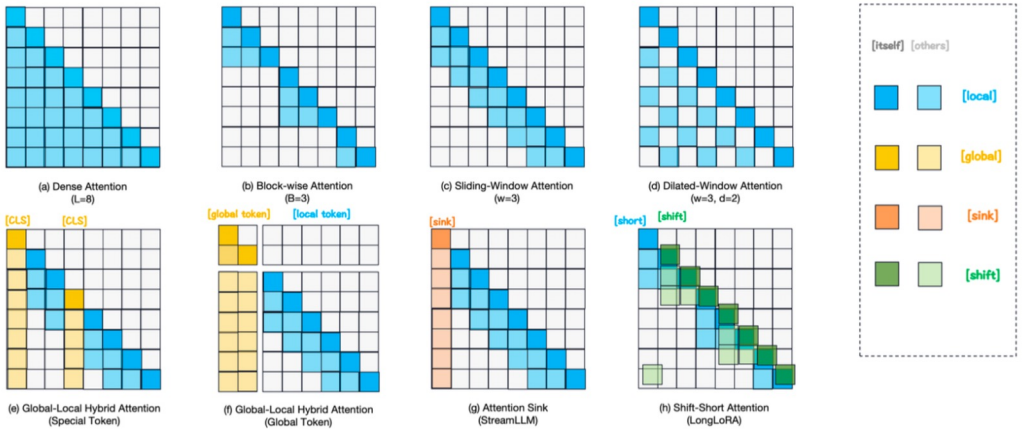


Fig. 2. The visualization of various typical local causal attention mechanisms. As the legend on the right indicates, tokens are distinguished by colors, with shades denoting attention to themselves (darker) or attention to the preceding others (lighter).

Enhances **effectively manage longer sequences with improved efficiency**

Hierarchical Attention

- **Goal:** Merge higher-level global information with lower-level local attention for efficient and scalable processing
- Impact
  - **Complexity Reduction**: Achieves sub-quadratic computational and memory costs while preserving the expressiveness of full attention
  - **Contextual Balance**: Maintains a balance between local and global context for inherent locality principle
- **Approaches**
  - **Two-Level Hierarchy**
    - Uses self-attention across two levels: word-to-sentence and sentence-to-document (e.g. HAN)
  - **Multi-Level Hierarchy**
    - Introduces fine-to-coarse attention via **binary partitioning**, formalizing as a graph neural network(e.g BPT**)**
    - Controls attention span with a soft attention mask (e.g. Adaptive Span Transformer)
  - **Advanced Hierarchical Mechanisms**
    - Partitions attention matrix into blocks with different low-rank ranges (e.g. H-Transformer-1D)
    - Combines full-attention approximation with structured factorization (e.g. Combiner)

Sparse Attention

- Exploring Sparsity in Attention Matrices
  - **Sparse Attention Mask**: Introduces sparsity in attention, allowing tokens to selectively attend to a subset of other tokens
  - **Benefits**: Enhances computational efficiency and maintains global context awareness
- Types
  - **Fixed Sparsity Patterns**
    - **Sparse Transformer**: Uses row-column factorized attention to reduce complexity to $O(\text{sqrt}(L)\cdot Ld)$
    - **LogSparse**: Employs an exponentially sparse attention pattern, leading to memory usage of $O(L(\log L)\text{^}2)$
  - **Adaptive Sparsity Patterns**
    - **Expire-Span**: Introduces learnable scalars to retain or expire tokens based on importance.
    - **Routing Transformer**: Leverages k-means to assign queries to the top-k relevant keys, simplifying attention to $O(L\cdot\text{sqrt}(L)\text{^}2)$
  - **Graph Sparsification**
    - **Star-Transformer**: Implements a star-shaped topology for localized attention with global context.
    - **BigBird**: Based on random graph theory, enables linear complexity with efficient attention mechanisms.

Sparse Attention

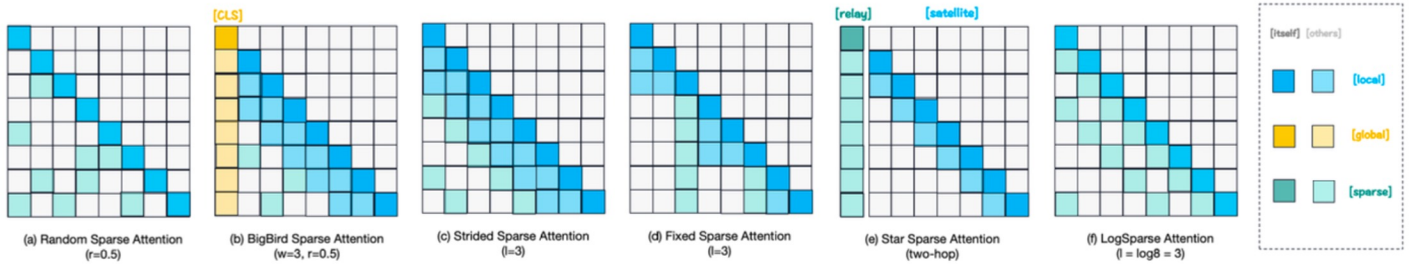


Fig. 3. The visualization of some typical causal sparse attention patterns. The legend on the right distinguishes token types based on their colors, where darker shades indicate attending to themselves while lighter ones represent attention to other previous tokens.

- Accommodates both long and short-range dependencies, scaling up to one billion tokens.
- Reduces dependency to a more manageable linear complexity

Approximated Attention

- **Goal**: Reduce the full attention computation by **leveraging sparsity and low-rankness** with linear complexity, optimizing precision trade-offs
- **Impact**: Provides sub-quadratic computation and memory complexity while maintaining the expressiveness of full attention
- **Techniques**
  - Low-Rank Approximation:
    - Linformer: Utilizes SVD for a low-rank approximation of the attention matrix, reducing complexity to $O(Lkd)$
  - Nested Attention:
    - Luna: Combines pack and unpack attention strategies to handle sequences of varying lengths without compromising parallelism
  - Kernelized Approximation:
    - Linear Transformer & Performer: Introduces **kernel-based attention** approximations, significantly cutting down on computational resources
  - **Hybrid Approaches**
    - **Sparse-Kernelized Hybrid**
    - **Scatterbrain**: combines **sparse matrices and kernelized feature maps** for enhanced efficiency and precision

IO-Aware Attention

- **Different**
  - Previous attention methods **trade off some attention quality for lower computation**
  - but IO-aware methods maintain exactness of attention calculations while **optimizing computational resources**
- **Offer exact attention computations with significantly reduced memory and time consumption**
  - A leap forward in the optimization of Transformer models for large-scale applications
- **Techniques**
  - **Memory-Efficient Attention:** Utilizes lazy softmax algorithm for numerically stable attention
  - **Flash Attention**: Achieves up to **7.6x speedup and 20x memory efficiency** with exact attention computation
  - **Paged Attention**
    - Addresses inference memory bottlenecks by **managing KV cache memory** with virtual memory paging techniques, improving efficiency and flexibility for batched requests
  - **Innovations and Improvements**
    - Sparse Clustered Factorization Attention: Extends Flash Attention to accommodate diverse sparsity patterns, **leading to 2 to 3.3 times training speedup**
    - Virtual Large Language Models: Proposes techniques to manage growing KV cache memory

Section 4: Long-Term Memory

in-context working memory -> The Transformer architecture often struggles with capturing long-term dependencies -> two main avenues to address: **(1) Internal MemoryCache; (2) External MemoryBank**

Internal MemoryCache trades space for time by using caching mechanisms to reduce computation. However, after model training is completed, it is difficult to update the internal knowledge, which is why such methods are rarely used nowadays.

Instead, the External Memory Bank method is mainly used.

Internal MemoryCache

- **Segment-Level Recurrence.**
  - Cache the output of m previous consecutive segments in the last layer and concatenate them into the current segment in the present layer to extend the context for the current query.
- **Retrospective Recurrence.**
  - Concatenate the output hidden states of previous segments in the same layer, instead of the last layer.
- **Continuous-Signal Memory.**
  - The ∞-former model uses a continuous signal representation to achieve unbounded long-term memory.

External MemoryBank

- **Cosine-Based Retrieval Criteria.**
  - LangChain is an open-source framework designed for chatbots, which processes local documentation into a memory bank using LLMs and retrieves context using cosine similarity to enhance interaction and response generation.
- **Heuristic Retrieval Criteria.**
  - Used for enhancing large language models with memory banks, enabling more efficient and context-aware data handling and retrieval in applications like chatbots and knowledge-based systems.
- **Learnable Retrieval Criteria.**
  - REALM use MLM to train a neural knowledge retriever
  - LongMem decouples the memory retrieval process using a SideNet.
  - FOT introduces a novel contrast training method to refine the key-value space and enhance retrieval accuracy as the size of the memory bank expands.

Section 5: Extrapolative PEs

— Extrapolative Positional Encodings. Current PEs play the undeniable role in length generalization in more general scenarios.

- **Enhancing Understanding**
  - Rethinking PEs as β-Encoding.
  - Length Extrapolation Dilemma.
- **Attention Bias**
  - As alternative mechanisms to explicitly encoding positional information, attention bias have been explored to capture the sequentiality and temporality of natural language incorporated into the attention kernel.
- **Extended RoPE**
  - Several research works have aimed to extend RoPE using various strategies to enhance its length extrapolation capabilities.
    - Scaling Strategies.
    - Truncation Strategies.
    - Rearrangement Strategies.

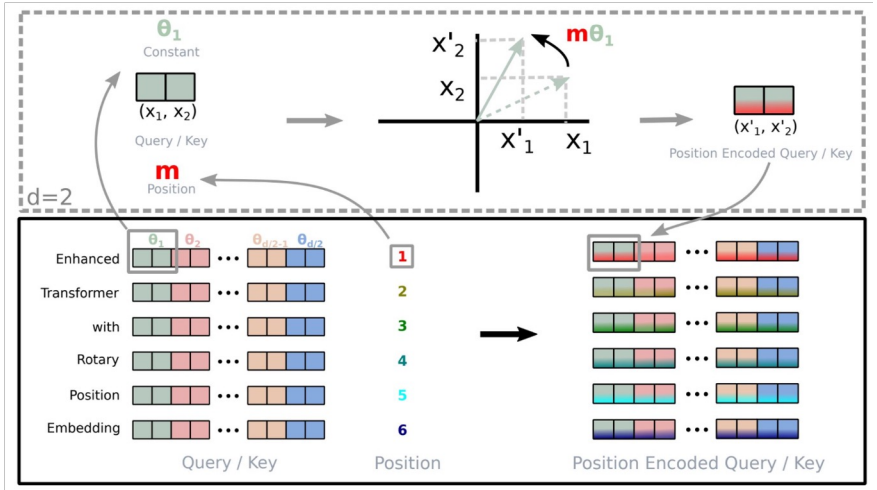


Figure 1: Implementation of Rotary Position Embedding(RoPE).

Section 6: Context Processing

- **Context Selection**
  - **Various strategies** employed by different models to effectively manage long text segments within the limited context window of LLMs, **involving segment partitioning, scoring based on selection criteria, and iterative or simultaneous selection processes to prioritize the most relevant segments for processing.**
- **Context Aggregation**
  - Extracting and integrating information from all context segments to generate a coherent final answer, through techniques like Fusion-in-Decoder, Map Reduce, Refinement.
  - Handling parallel context windows, each with different strategies for encoding, merging, and refining the information from multiple segments.
- **Context Compression**
  - Methods for compressing long contexts to fit within the maximum sequence length constraints of LLMs.
    - Soft compression: create condensed and abstract representations through embedded learning.
    - Hard Compression: eliminate redundancies using metrics like self-information and perplexity to optimize input quality before processing.

Section 7: Miscellaneous Solution

Not be exhaustive or specific to Transformer-based models. Many of these techniques are applicable universally to any model equipped with deep neural networks, albeit particularly crucial for large-scale LLMs.

- **Specific Objectives**
  - Recent research explores tailored approaches to adapt pretraining for specific tasks, aiming to enhance LLMs' effectiveness in capturing intricate long-range dependencies and discourse structures in longer texts compared to shorter ones. (XLNet, ERNIE-Doc, DANCE, PEGASUS, PRIMERA)
- **Mixture of Experts**
  - Mixture of Experts (MoE) enhances large language models by incorporating specialized expert modules and dynamic gating mechanisms to optimize task performance, reduce computational demands, and improve efficiency and effectiveness in handling large-scale contexts.
- **Parallelism**
  - Leveraging modern aggregated GPU memory within and across nodes, recent research has introduced various parallelism strategies to scale up model sizes and extend sequence length, including Data Parallelism (DP), Tensor Parallelism (TP), Pipeline Parallelism (PP), Sequence Parallelism (SP), Expert Parallelism (EP).
- **Weight Compression**
  - Various methods enhance memory efficiency in large-scale LLMs through weight compression techniques, including pruning, factorization, quantization, partitioning, and distillation.

Section 8: Evaluation Necessity & Optimization Toolkit

- Explore evaluation necessities for assessing long-context capabilities of LLMs, including datasets, metrics, and baseline models.
- Investigate popular optimization toolkits, such as libraries, frameworks, and compilers, to enhance LLM efficiency and effectiveness during development.

Table 1. Basic information about existing datasets specific for various NLP tasks with long-text inputs.

Dataset	Language	Task Amount	Task Types										Lengths (kilo words)			Quality		Splits	Count	Format
			LM	*MCQA*	*ExtQA*	*Summ*	*Class*	*Match*	*Math*	*Code*	*OpenW*	*MT*	*Avg*	*Min*	*Max*	*Human Labeled*	*Model Assisted*			
ArXiv + PubMed	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	5.2	0	157.3	✓	✗	train/test/val	322K/13.1K/13.1K	jsonl
BigPatent	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	3.2	0.2	83.2	✓	✗	train/test/val	1.2M/67.1K/67.1K	json
BookSum	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	4.5	0.04	115.8	✓	✗	train/test/val	9.6K/1.4K/1.5K	csv
CAIL2019-SCM	zh	1	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	2.0	1.8	2.6	✓	✗	train/test/val	5.1K/1.5K/1.5K	jsonl
ChapterBreak	en	1	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	25.4	2.3	405.8	✓	✗	train	9.6K	json
CNN/DailyMail	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.8	0	2.9	✓	✗	test	312K	txt
ContractNLI	en	1	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	2.0	0.5	8.7	✓	✗	train/test/dev	423/123/61	json
DuLeMon	zh	1	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.6	0.3	1.4	✓	✗	train/test/dev	25.4K/1.1K/1.1K	jsonl
ECtHR	en	1	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	2.2	0.01	51.3	✓	✗	train/test/dev	7.3K/3K/1.3K	jsonl
GovReport	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	43.5	0.2	1386.2	✓	✗	test	19.4K	json
HotpotQA	en	1	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	0.9	0.01	2.0	✓	✗	train/dev	90K/14.8K	json
InfiniteBench	en/zh	12	✗	✓	✓	✓	✗	✗	✓	✓	✗	✗	71.1	0.1	560.3	✓	✗	test	3.9K	jsonl
LCC-Python	py	1	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	1.4	0.2	23.3	✓	✗	train/test/val	100K/10K/10K	parquet
LEval	en	20	✗	✓	✓	✓	✗	✗	✓	✓	✓	✗	9.2	2.0	137.5	✓	✓	test	537	jsonl
LongAlpaca	en	1	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	6.7	0	32.7	✓	✗	train	12K	json
LongBench	en/zh	21	✗	✗	✓	✓	✓	✓	✗	✓	✗	✗	7.2	0.1	44.2	✓	✓	test	8.4K	jsonl
LongChat-Lines	en	1	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	2.6	0.6	5.6	✓	✗	test	700	parquet
LOT	zh	4	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	0.2	0.06	0.5	✓	✗	train/test/dev	35.2K/2.4K/1.8K	jsonl
LRA - AAN	en	1	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	4.7	0.02	55.5	✓	✗	train/test/dev	147K/17.4K/18K	tsv
LRA - ListOps	en	1	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	3	0.01	5.9	✓	✗	train/test/dev	96K/2K/2K	tsv
MuLD	en	6	✗	✗	✓	✓	✓	✗	✗	✗	✗	✓	27.7	0	359.1	✓	✗	train/test/val	155.9K/14.4K/11.6K	jsonl
MultiNews	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	2.1	0.1	464.2	✓	✗	train/test/val	45.0K/5.6K/5.6K	txt
Multi-Session Chat	en	1	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	0.3	0.1	1.2	✓	✗	train/test/val	17.9K/2.5K/3K	parquet
Nature Questions	en	1	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	9.8	0.2	169.3	✓	✗	train/dev	307K/7.8K	json
NewsGroups	en	1	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	0.3	0	11.8	✓	✗	test	20K	txt
NewsRoom	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.7	0	178.5	✓	✗	train/test/dev	995.0K/108.9K/108.8K	jsonl
OpenChat-ShareGPT4-Clean	en	1	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	1.6	0	152.8	✓	✓	train	80.2K	json
ProofNet	en	1	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	0.2	0.05	0.7	✓	✗	test/val	186/185	jsonl
QMSum	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	10.8	1.7	26.8	✓	✗	train/test/val	162/35/35	jsonl
SCROLLS	en	7	✗	✓	✓	✓	✗	✗	✓	✗	✗	✗	33.0	0.2	356.1	✓	✗	train/test/val	89.7K/17.5K/12.3K	jsonl
SQuAD	en	1	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	0.1	0.02	0.7	✓	✗	train/val	87.6K/10.6K	parquet
SummScreen	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	7.3	1.6	24.0	✓	✗	train/test/dev	22.6K/2.1K/2.1K	jsonl
Synthetic-Persona-Chat	en	1	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	0.4	0.05	0.8	✓	✓	train/test/val	8.9K/968/1K	csv
THUCnews	zh	1	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	0.9	0	79.5	✓	✗	test	836K	txt
UltraChat	en	1	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	1.0	0.03	3.6	✓	✓	train	1.4M	jsonl
WikiQA-AlteredNumericQA	en	1	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	4.0	0.8	11.2	✓	✗	test	1.8K	parquet
WikiQA-FreeFormQA	en	1	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	3.8	0.6	11.5	✓	✗	test	2.4K	parquet
WMT14 EN-CS	en/cs	1	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	0.04	0	3.6	✓	✗	train/test/cal	1M/3K/3K	sgm
XSum	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.4	0	29.2	✓	✗	train/test/val	204K/11.3K/11.3K	summary

- **Datasets**
  - detailed information on each dataset is available, covering language, task types, length statistics, quality, splits, count and format.

Table 2. Some common metrics adopted for evaluation on each specific NLP task type as depicted in Appendix. A.

Task Types	Metric Types								
	CE/PPL	*BPC/BPW*	*Acc/F1*	*EM*	*ROUGE-1/-2/-L*	*BLEU/METEOR/TER*	*EntMent*	*Pass@k*	*Human/Model Judge*
LM	✓	✓	✓	✗	✗	✗	✗	✗	✓
MCQA	✗	✗	✓	✗	✗	✗	✗	✓	✗
ExtQA	✗	✗	✓	✓	✓	✓	✓	✗	✓
Summ	✗	✗	✗	✗	✓	✓	✓	✗	✓
Class	✗	✗	✓	✗	✗	✗	✗	✗	✗
Match	✗	✗	✓	✗	✗	✗	✗	✓	✗
Math	✗	✗	✓	✓	✗	✗	✗	✓	✓
Code	✗	✗	✓	✓	✓	✓	✓	✓	✓
OpenW	✗	✗	✓	✓	✓	✓	✓	✓	✓
MT	✗	✗	✗	✗	✗	✓	✓	✓	✓

Note: The ✗ in the table does not imply that a specific metric cannot be applied to a task. Rather, it suggests that the metric might be less commonly used or that there could be more suitable alternatives.

- **Metrics**

a summary of nine categories of general evaluation metrics commonly employed across ten NLP task types, encompassing language modeling, question answering, summarization, math solving, code generation, and open-ended writing, among others.

Table 3. Basic information for some long-context models widely-used as baselines.

Model	Open Source	Base	Main Usage	Main Lang	L_{max} (k)	Param Size (B)	Mem Occ (GB)	Disk Occ (GB)	Links
Anima-7B-100k	✓	Llama2	chat	zh	100	6.7	12.6	12.6	hf | github
ChatGLM2-6B-32k	✓	GLM	chat	zh	32	6.2	11.7	11.6	hf | github
ChatGLM3-6B-32k	✓	GLM	chat	zh	32	6.2	11.7	11.6	hf | github
Chinese-Alpaca2-7B-16k	✓	Llama2	instruct	zh	16	6.9	25.9	12.9	hf | github
Chinese-Llama2-7B-16k	✓	Llama2	chat	zh	16	6.9	26.3	12.9	hf | github
Chinese-Mixtral	✓	Mixtral	chat	zh	32	46.7	175.0	87.0	hf | github
Chinese-Mixtral-Instruct	✓	Mixtral	instruct	zh	32	46.7	175.0	87.0	hf | github
Claude2	✗	Claude	chat	en	100	?	?	?	acc | home
CodeLlama-7B	✓	Llama2	code	py	16	6.7	25.6	12.6	hf | home | paper
CodeLlama-13B	✓	Llama2	code	py	16	13.0	49.1	24.2	hf | home | paper
CodeLlama-34B	✓	Llama2	code	py	16	33.7	126.5	62.9	hf | home | paper
Giraffe-13B-32k-v3	✓	Llama2	instruct	en	32	13.0	48.6	24.2	hf | github | paper
Giraffe-v2-70B-32k	✓	Llama2	instruct	en	32	69.0	227.4	128.5	hf | github | paper
GPT3.5-Turbo-16k	✗	GPT3	chat	en	16	?	?	?	acc | home | doc
GPT4	✗	GPT4	chat	en	8	?	?	?	acc | home | doc
GPT4-32k	✗	GPT4	chat	en	32	?	?	?	acc | home | doc
GPT4-Turbo	✗	GPT4	chat	en	128	?	?	?	acc | home | doc
InternLM-Chat-7B	✓	Llama2	chat	en	200	6.7	12.6	12.6	hf | github
Llama2-7B-32k	✓	Llama2	chat	en	32	6.7	12.6	12.6	hf | home
Llama2-7B-Instruct-32k	✓	Llama2	instruct	en	32	6.7	12.6	12.6	hf | home
LLongMA2-7B-16k-flash	✓	Llama2	chat	en	16	6.7	12.6	12.6	hf | paper
LongChat-v1.5-7B-32k	✓	Llama2	chat	en	32	6.7	12.6	12.6	hf | github | blog
Mistral-7B-v0.1	✓	Mistral	chat	en	32	7.2	28.0	13.5	hf | paper
Mistral-7B-Instruct-v0.2	✓	Mistral	instruct	en	32	7.2	28.0	13.5	hf | paper
Mixtral-8x7B-v0.1	✓	Mixtral	chat	en	32	46.7	175.0	87.0	hf | blog
Mixtral-8x7B-Instruct-v0.1	✓	Mixtral	instruct	en	32	46.7	175.0	87.0	hf | blog
MPT-7B-Storywriter	✓	MPT	gen	en	65	6.6	12.4	12.4	hf | blog
NeuralChat-7B-v3.1	✓	Mistral	chat	en	32	7.2	28.0	13.5	hf | blog
OpenHermes2.5-7B	✓	Mistral	chat	en	32	7.2	28.0	13.5	hf | github
QWen-7B	✓	QWen	chat	zh	32	7.7	14.4	14.4	hf | paper
Vicuna-v1.5-7B-16k	✓	Llama2	chat	en	16	6.7	12.6	12.6	hf | github | blog
WizardCoder-Python-7B-v1.0	✓	Llama2	code	py	16	6.7	12.8	12.6	hf | github
WizardMath-7B-v1.1	✓	Mistral	math	en	32	7.2	14.0	13.5	hf | github
XGen-7B-Instruct-8k	✓	Llama2	instruct	en	8	6.7	12.6	12.6	hf | paper

- **Baselines**
  - Gather a list of pretrained/finetuned LLMs commonly, serving as baselines for evaluating long-context capabilities across various downstream tasks.

Table 4. The toolkits summary for enhancing LLMs efficiency and effectiveness across different stages.

Toolkit	Type	Utilities for Stages			
		Pretraining	*Finetuning*	*Inference*	*Application*
Accelerate	library	Integrate TorchRun, FSDP DeepSpeed, Megatron-LM Local SGD			CPU/GPU/ TPUs/Apple Silicon Auto Device Management
AutoGen	framework			Multi-Config Inference	Multi-Agent Conversation
BitsandBytes	library	8bit Optimizers 8bit Matrix Multiplication QLoRA		QLoRA	4bit/8bit Quantization
Colossal-AI	library	Integrate DP, PP, 1D/2D/2.5D/3D TP, ZERO Auto Parallelism			
DeepSpeed	framework	DP, PP, TP, ZERO, Offload, Sparse Attention Kernel			
DeepSpeed-MII	framework			Dynamic SplitFuse	
FlashAttention	library	Kernel-Fused Flash-Attention			
HuggingFace TGI	system			TP Optimized Architectures Continuous Batching Quantization	
LangChain	framework				Prompt Management Memory Management Agent Management
LangChain-Chatchat	framework				Integrate Langchain RAG
Llama-Factory	library		Integrate LoRA, QLoRA, PPO, DPO, Reward Modeling		
Megatron-LM	framework	DP, PP, SP, ZERO Activation Checkpointing			
Orca	system			Iteration-level scheduling Selective Batching	
PEFT	library	LoRA, Prefix-Tuning, Prompt-Tuning			
Petals	framework		Multi-Party Distributed Collaboration		
PrivateGPT	framework				Private RAG
Pytorch FSDP	framework	Fully Sharded Data Parallel			
Pytorch SDPA	function	Integrate Flash-Attention, Memory-Efficient Attention			
TensorRT-LLM	library			Python API for TensorRT Engines In-flight Batching	
Triton	compiler	Python API for GPU Kernels			
vLLM	library			Paged Attention	
xFormers	library	Memory-Efficient Attention			

- **Toolkit**
  - Collect a diverse array of valuable toolkits to optimize the efficiency and effectiveness of LLMs across their development lifecycle.

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

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré

Presented by Kefan Song (ks8vf)

Motivation1: Modeling Longer Sequences

NLP:

Large context required to read books, plays and instruction manuals

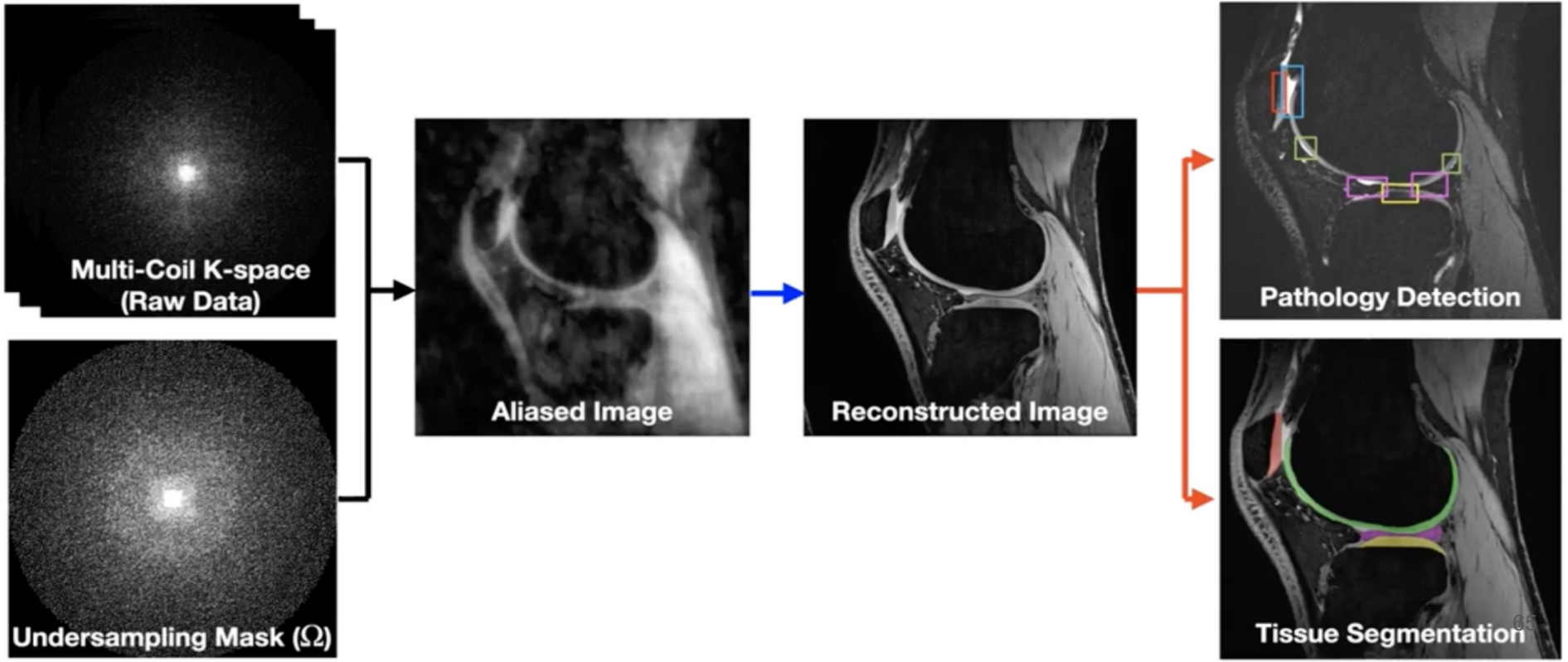
Computer vision:

Higher resolution image requires longer sequence

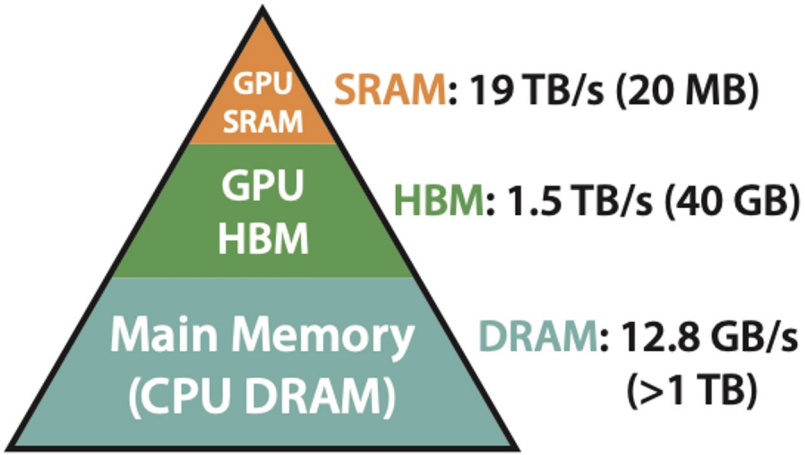
Sequence Data:

Time-series, audio, video, medical imaging

High Resolution MRI: Detection and Segmentation



Motivation2: Attention is bottlenecked by I/O from HBM



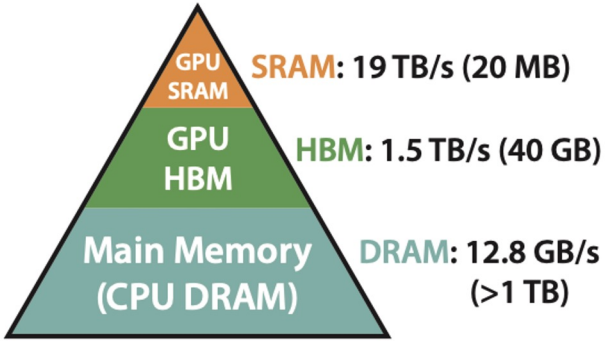
Memory Hierarchy with Bandwidth & Memory Size

Motivation2: Attention is bottlenecked by I/O from HBM

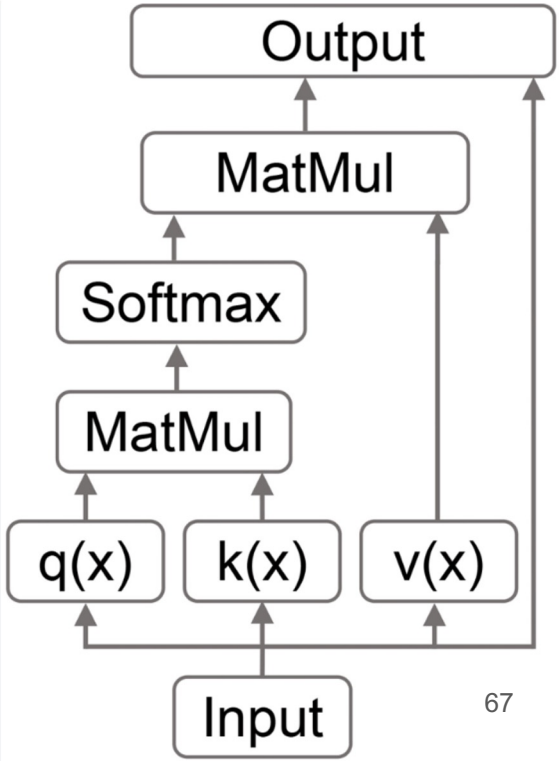
Algorithm 0 Standard Attention Implementation

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM.

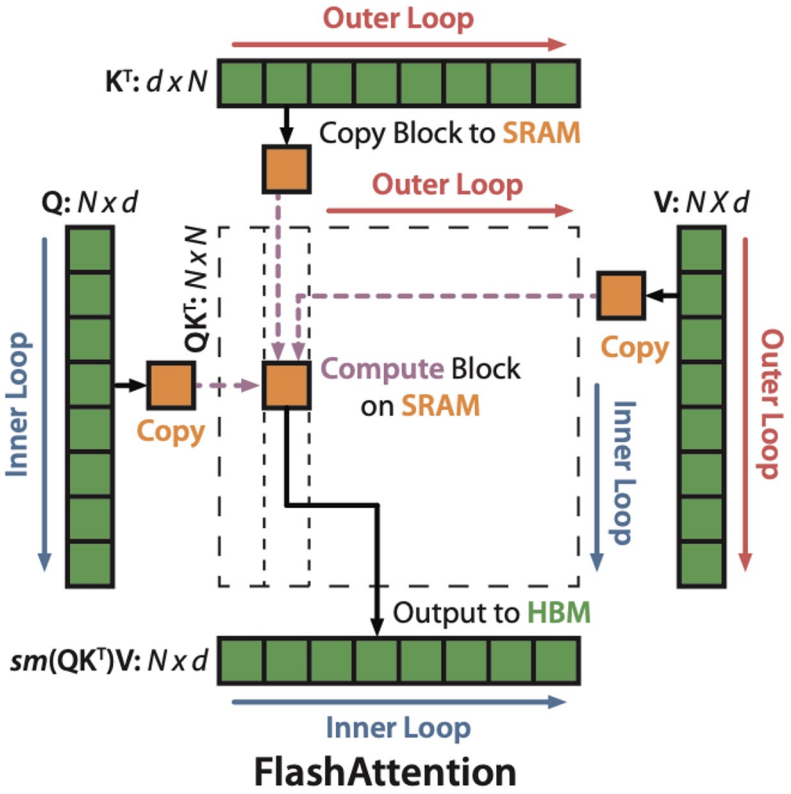
1: Load $\mathbf{Q}, \mathbf{K}$ by blocks from HBM, compute $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top$, write $\mathbf{S}$ to HBM.
2: Read $\mathbf{S}$ from HBM, compute $\mathbf{P} = \mathrm{softmax}(\mathbf{S})$, write $\mathbf{P}$ to HBM.
3: Load $\mathbf{P}$ and $\mathbf{V}$ by blocks from HBM, compute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



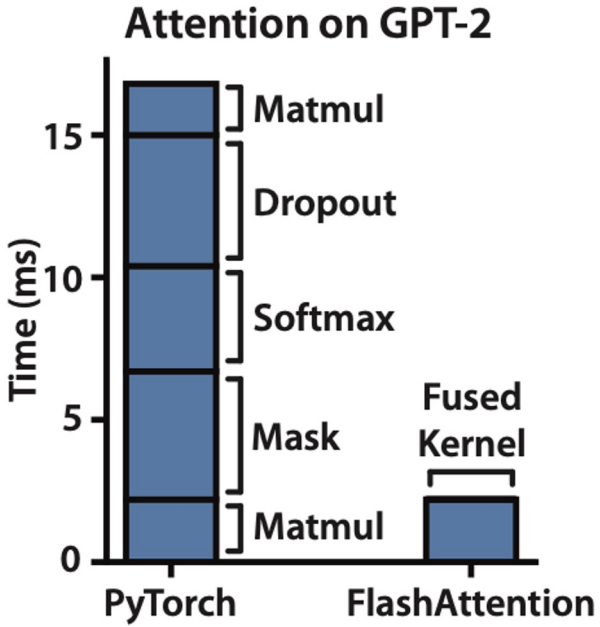
Memory Hierarchy with Bandwidth & Memory Size



FlashAttention



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 $\mathbf{O}$.

"From Online Softmax to FlashAttention" a Note by Zihao Ye

Safe Softmax

$$\frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} = \frac{e^{x_i - m}}{\sum_{j=1}^{N} e^{x_j - m}}$$

"From Online Softmax to FlashAttention" a Note by Zihao Ye

Safe Softmax

$$\frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} = \frac{e^{x_i - m}}{\sum_{j=1}^{N} e^{x_j - m}}$$

Algorithm 3-pass safe softmax

NOTATIONS
$\{m_i\}$: $\max_{j=1}^{i}\{x_j\}$, with initial value $m_0 = -\infty$.
$\{d_i\}$: $\sum_{j=1}^{i} e^{x_j - m_N}$, with initial value $d_0 = 0$, d_N is the denominator of safe softmax.
$\{a_i\}$: the final softmax value.

BODY
for $i \leftarrow 1, N$ **do**

$$m_i \leftarrow \max(m_{i-1}, x_i) \tag{7}$$

end
for $i \leftarrow 1, N$ **do**

$$d_i \leftarrow d_{i-1} + e^{x_i - m_N} \tag{8}$$

end
for $i \leftarrow 1, N$ **do**

$$a_i \leftarrow \frac{e^{x_i - m_N}}{d_N} \tag{9}$$

end

Online Softmax

Algorithm 2-pass online softmax

for $i \leftarrow 1, N$ **do**

$$m_i \leftarrow \max\left(m_{i-1}, x_i\right)$$
$$d'_i \leftarrow d'_{i-1}\, e^{m_{i-1}-m_i} + e^{x_i-m_i}$$

end

for $i \leftarrow 1, N$ **do**

$$a_i \leftarrow \frac{e^{x_i-m_N}}{d'_N}$$

end

FlashAttention Algorithm

Decompose softmax into smaller softmax

Algorithm FlashAttention

for $i \leftarrow 1, N$ **do**

$$
\begin{aligned}
x_i &\leftarrow Q[k,:]\, K^T[:,i] \\
m_i &\leftarrow \max(m_{i-1}, x_i) \\
d'_i &\leftarrow d'_{i-1}\, e^{m_{i-1}-m_i} + e^{x_i - m_i} \\
\boldsymbol{o}'_i &\leftarrow \boldsymbol{o}'_{i-1} \frac{d'_{i-1}\, e^{m_{i-1}-m_i}}{d'_i} + \frac{e^{x_i-m_i}}{d'_i} V[i,:]
\end{aligned}
$$

end

$$O[k,:] \leftarrow \boldsymbol{o}'_N$$

FlashAttention Algorithm

BODY

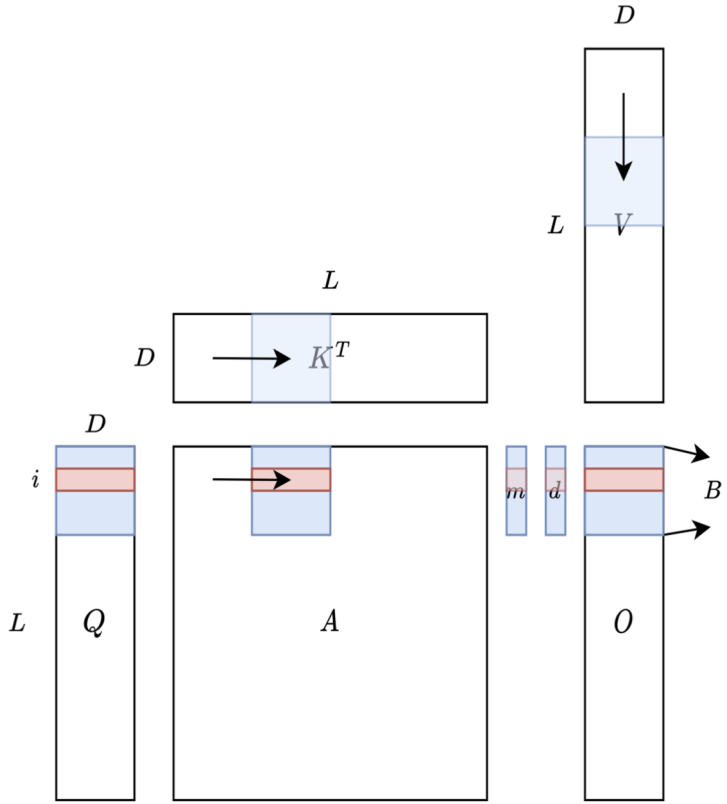
for $i \leftarrow 1, \#\text{tiles}$ **do**

$$
\begin{aligned}
\boldsymbol{x}_i &\leftarrow Q[k,:]\, K^T[:, (i-1)\,b{:}\,i\,b] \\
m_i^{(\text{local})} &= \max_{j=1}^{b} (\boldsymbol{x}_i[j]) \\
m_i &\leftarrow \max\left(m_{i-1}, m_i^{(\text{local})}\right) \\
d_i' &\leftarrow d_{i-1}'\, e^{m_{i-1}-m_i} + \sum_{j=1}^{b} e^{\boldsymbol{x}_i[j]-m_i} \\
\boldsymbol{o}_i' &\leftarrow \boldsymbol{o}_{i-1}' \frac{d_{i-1}'\, e^{m_{i-1}-m_i}}{d_i'} + \sum_{j=1}^{b} \frac{e^{\boldsymbol{x}_i[j]-m_i}}{d_i'} V[j+(i-1)\,b,:]
\end{aligned}
$$

end

$$O[k,:] \leftarrow \boldsymbol{o}_{N/b}'$$

FlashAttention



Evaluation: Faster Training Speed

Table 1: Training time of BERT-large, starting from the same initialization provided by the MLPerf benchmark, to reach the target accuracy of 72.0% on masked language modeling. Averaged over 10 runs on 8×A100 GPUs.

BERT Implementation	Training time (minutes)
Nvidia MLPerf 1.1 [58]	20.0 ± 1.5
FLASHATTENTION (ours)	**17.4** ± 1.4

Evaluation: Language Modeling with Long Context

Model implementations	Context length	OpenWebText (ppl)	Training time (speedup)
GPT-2 small - Megatron-LM	1k	18.2	4.7 days (1.0×)
GPT-2 small - FLASHATTENTION	1k	18.2	**2.7 days (1.7×)**
GPT-2 small - FLASHATTENTION	2k	17.6	3.0 days (1.6×)
GPT-2 small - FLASHATTENTION	4k	**17.5**	3.6 days (1.3×)

Evaluation: Long Context Image Classification

Model	Path-X	Path-256
Transformer	✗	✗
Linformer [84]	✗	✗
Linear Attention [50]	✗	✗
Performer [12]	✗	✗
Local Attention [80]	✗	✗
Reformer [51]	✗	✗
SMYRF [19]	✗	✗
FlashAttention	**61.4**	✗
Block-sparse FlashAttention	56.0	**63.1**

Evaluation: Attention Benchmark

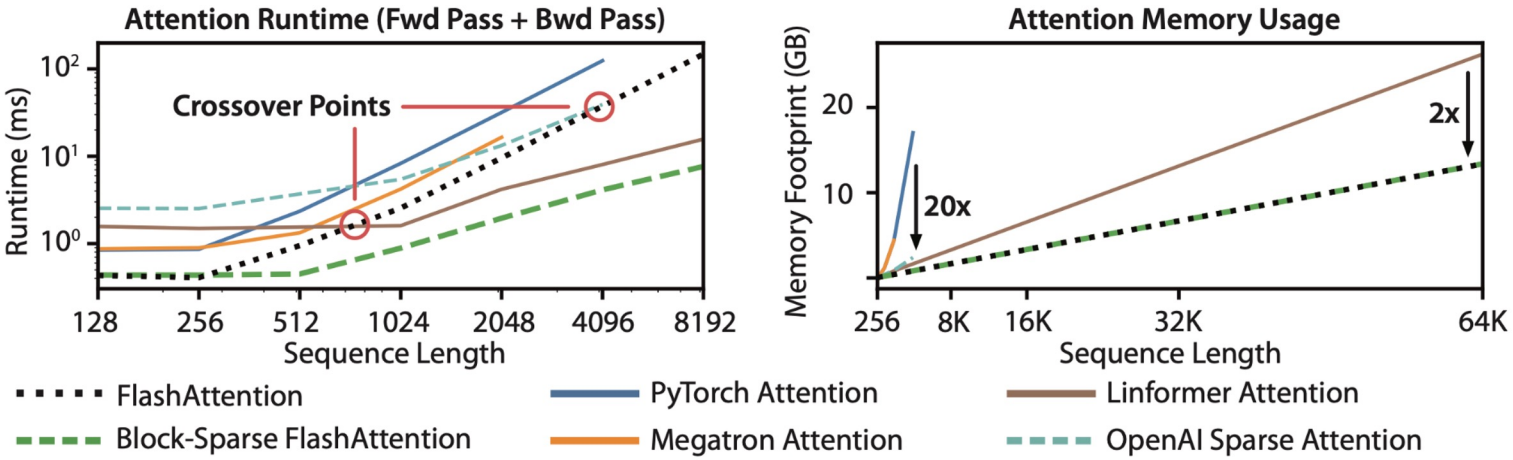


Figure 3: **Left:** runtime of forward pass + backward pass. **Right:** attention memory usage.

Thank you