Gated Slot Attention for Efficient Linear-Time Sequence Modeling

Yu Zhang🦙 Songlin Yang🦙 Ruijie Zhu Yue Zhang Leyang Cui Yiqiao Wang Bolun Wang Freda Shi Bailin Wang Wei Bi Peng Zhou Guohong Fu

https://github.com/sustcsonglin/flash-linear-attention 🤗 https://huggingface.co/fla-hub

yzhang.cs@outlook.com yangsl66@mit.edu 🦙 Equal contribution

Model summary

This work introduces Gated Slot Attention (GSA), enhancing Attention with Bounded Memory Control (ABC, Peng et al., 2022) with **gating mechanisms** and a **hardware-efficient implementation** for **larger-scale language modeling**.

Model	Self Attention	ABC	GSA
Output	$\mathbf{o}_t = \widetilde{\mathbf{V}}^T \mathrm{softmax}(\widetilde{\mathbf{K}}_t^T \mathbf{q}_t) \in \mathbb{R}^d$		
Key update	$\widetilde{\mathbf{K}}_t = [\widetilde{\mathbf{K}}_{t-1}; \mathbf{k}_t]$	$\widetilde{\mathbf{K}}_t = \widetilde{\mathbf{K}}_{t-1} + \boldsymbol{\phi}_t \otimes \mathbf{k}_t$	$\widetilde{\mathbf{K}}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t)\widetilde{\mathbf{K}}_{t-1} + (1-\alpha_t) \otimes \mathbf{k}_t$
Key size	Linear $(t \times d)$	Constant $(m \times d)$	Constant $(m \times d)$
Value update	$\widetilde{\mathbf{V}}_t = [\widetilde{\mathbf{V}}_{t-1}; \mathbf{v}_t]$	$\widetilde{\mathbf{V}}_t = \widetilde{\mathbf{V}}_{t-1} + \boldsymbol{\phi}_t \otimes \mathbf{v}_t$	$\widetilde{\mathbf{V}}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t)\widetilde{\mathbf{V}}_{t-1} + (1-\alpha_t) \otimes \mathbf{v}_t$
Value size	Linear $(t \times d)$	Constant $(m \times d)$	Constant $(m \times d)$

Table 1. Comparison of Different Attention Mechanism Update Rules

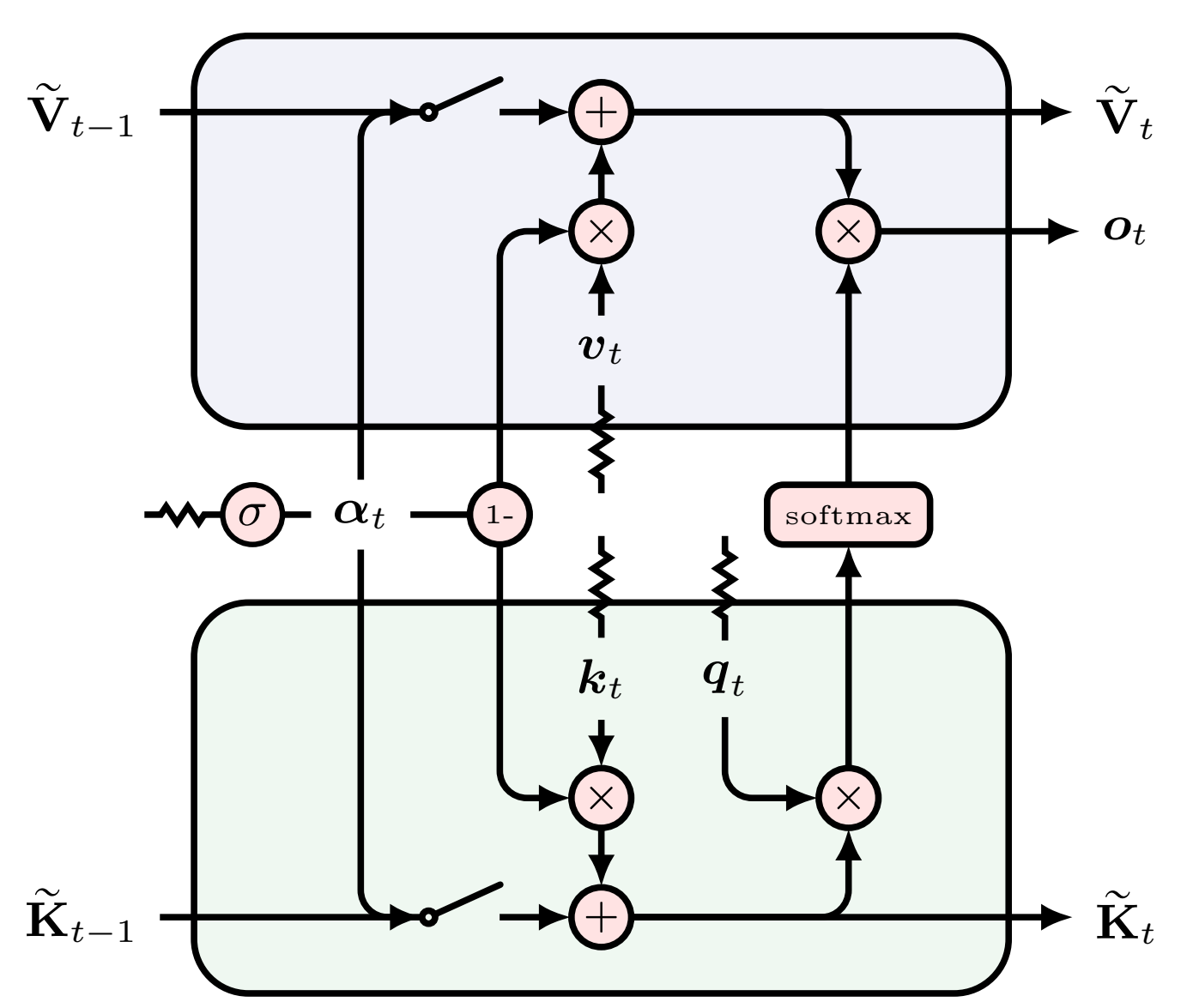
- Attention has **unbounded memory size**: **quadratic** time complexity and **linear** space complexity.
- ABC and GSA operate with a **fixed memory size**: **linear** time complexity and **constant** space complexity.
- GSA demonstrates improved **state efficiency**, achieving comparable or superior performance with a smaller state size, even in **recall-intensive tasks**. A smaller state size is critical for enhancing **inference efficiency**.
- GSA **outperforms** ABC in language modeling by a large margin thanks to the **gating mechanism**.
- GSA retains the **softmax** operator, making them well-suited for **"fine-tuning pretrained transformers to RNNs"** scenarios, thereby **reducing the cost of training from scratch**.

ABC and GSA as two-pass (gated) linear attention

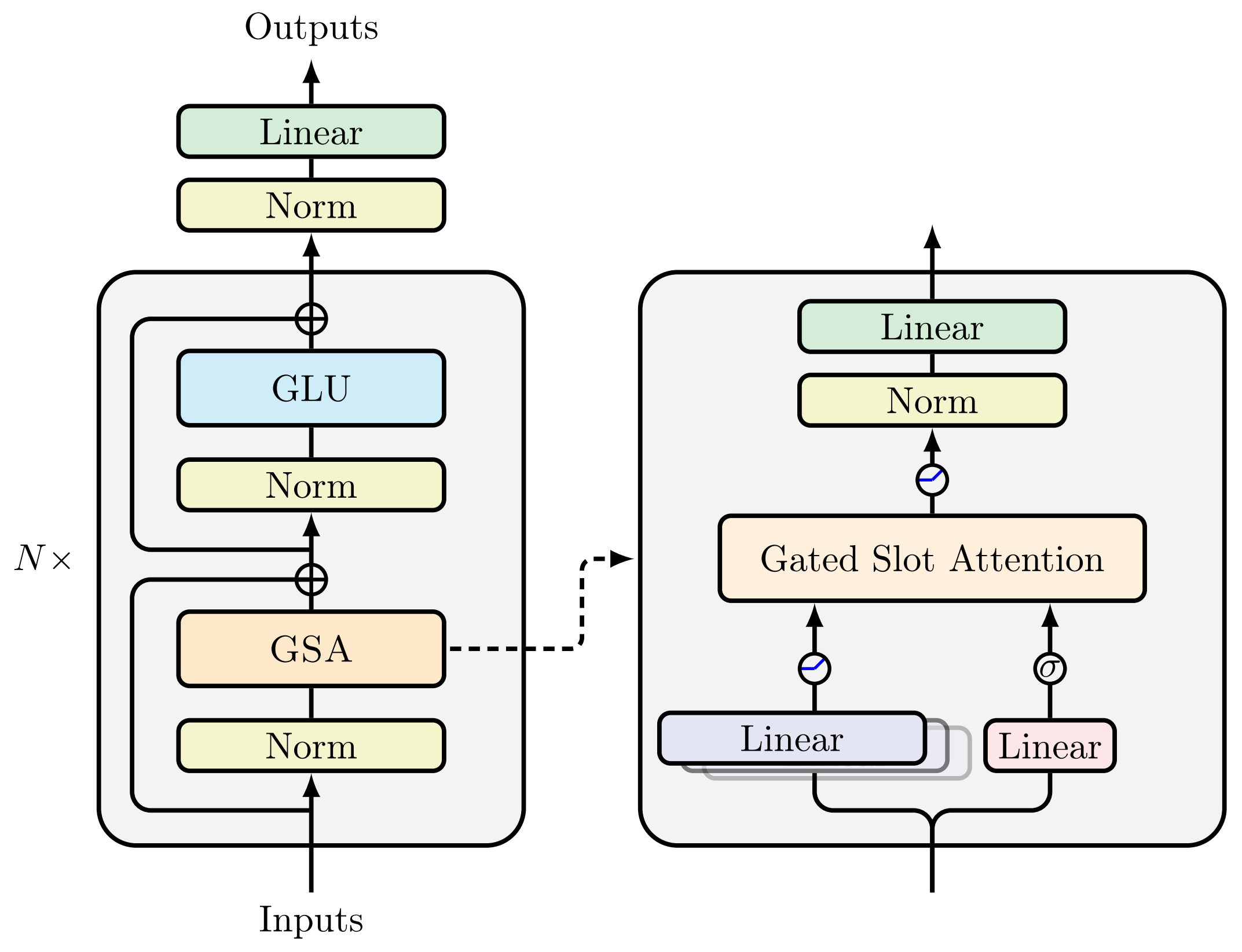
Definitions	
Linear Attention (LA)	$\mathrm{LA}(\{\boldsymbol{q}_i, \boldsymbol{k}_i, \boldsymbol{v}_i\}_{i=1}^T)$: $\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{k}_t \otimes \boldsymbol{v}_t \in \mathbb{R}^{d\times d}$ $\boldsymbol{o}_t = \mathbf{S}_t^T \boldsymbol{q}_t \in \mathbb{R}^d$
Gated Linear Attention (GLA)	$\mathrm{GLA}(\{\boldsymbol{q}_i, \boldsymbol{k}_i, \boldsymbol{v}_i, \boldsymbol{\alpha}_i, \boldsymbol{\beta}_i\}_{i=1}^T) = \{\boldsymbol{o}_i\}_{i=1}^T$: $\mathbf{S}_t = \mathbf{G}_t \odot \mathbf{S}_{t-1} + \boldsymbol{k}_t \otimes \boldsymbol{v}_t \in \mathbb{R}^{d\times d}, \quad \mathbf{G}_t = \boldsymbol{\alpha}_t \otimes \boldsymbol{\beta}_t \in \mathbb{R}^{d\times d}$ $\boldsymbol{o}_t = \mathbf{S}_t^T \boldsymbol{q}_t \in \mathbb{R}^d$
Two-Pass Forms	
ABC	$\{\boldsymbol{o}'_i\}_{i=1}^T = \mathrm{LA}(\{\boldsymbol{q}_i, \boldsymbol{k}_i, \boldsymbol{\phi}_i\}_{i=1}^T)$ $\{\boldsymbol{o}_i\}_{i=1}^T = \mathrm{LA}(\{\mathrm{softmax}(\boldsymbol{o}'_i), \boldsymbol{\phi}_i, \boldsymbol{v}_i\}_{i=1}^T)$
GSA	$\{\boldsymbol{o}'_t\}_{t=1}^T = \mathrm{GLA}(\{\boldsymbol{q}_t, \boldsymbol{k}_t, 1-\boldsymbol{\alpha}_t, \boldsymbol{\alpha}_t, \mathbf{1}\}_{t=1}^T)$ $\{\boldsymbol{o}_t\}_{t=1}^T = \mathrm{GLA}(\{\mathrm{softmax}(\boldsymbol{o}'_t), 1-\boldsymbol{\alpha}_t, \boldsymbol{v}_t, \mathbf{1}, \boldsymbol{\alpha}_t\}_{t=1}^T)$

We can use the **flash-linear-attention** implementations for hardware-efficient training!

Model Architecture

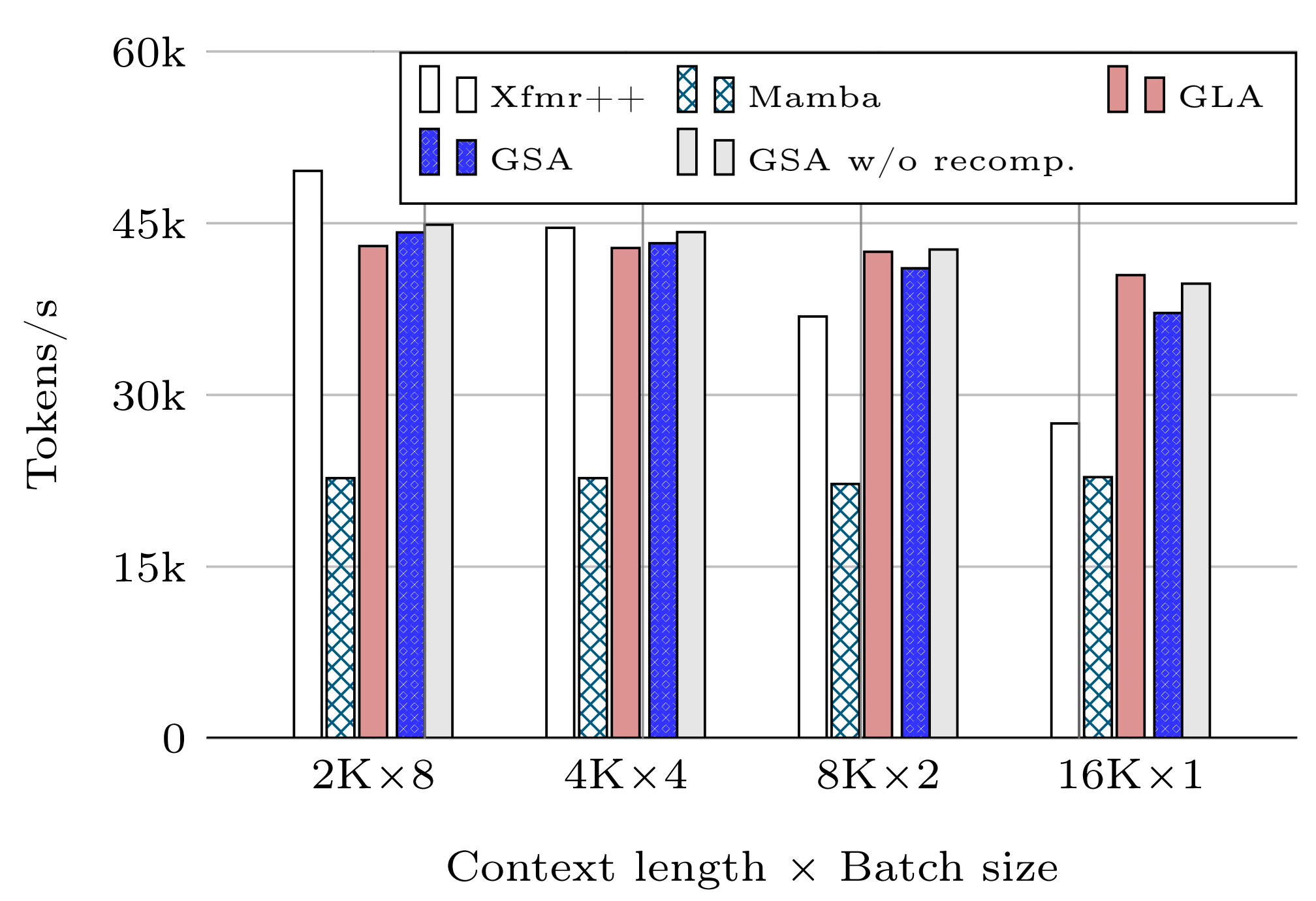

$\widetilde{\mathbf{V}}_{t-1}$
$\widetilde{\mathbf{V}}_t$
$\boldsymbol{o}_t$
$\boldsymbol{v}_t$
σ
$\boldsymbol{\alpha}_t$
1-
softmax
$\boldsymbol{k}_t$
$\boldsymbol{q}_t$
$\widetilde{\mathbf{K}}_{t-1}$
$\widetilde{\mathbf{K}}_t$

(a) The recurrent representation of GSA. $\rightsquigarrow$ means taking $\boldsymbol{x}_t$ as input.

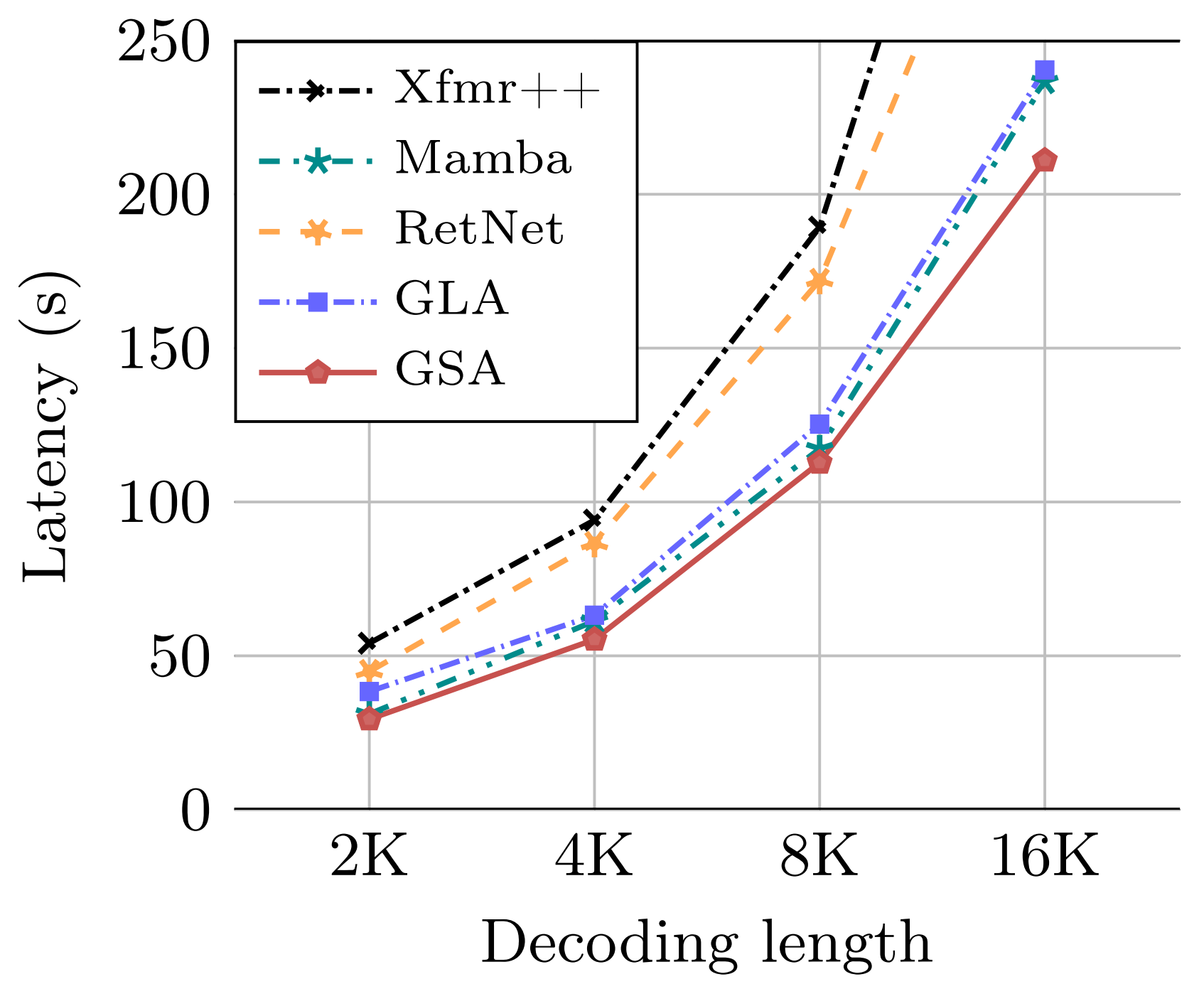


(b) The backbone of our proposed GSA models.

Efficiency



(a) Training throughputs (K tokens/s).



(b) Inference latencies on a single H100.

Ablation study

Table 2. Ablation study results for 340M models trained on 10B Slimpajama tokens.

Gating & Slots		*Non-linearity*	
GSA w/ 64 slots	13.51	$-$ softmax	14.03
w/o decay (ABC)	16.94	$-$ softmax $+$ Swish	13.71
w/ data-independent decay	15.83	$-$ softmax $+$ ReLU	13.69
w/ 32 slots	13.74	$-$ softmax $+$ ReLU^2	13.95
w/ 128 slots	**13.46**		

Language Modeling and Common-Sense Reasoning Performance

	State size	Lamb. ppl$_\downarrow$	Wiki. ppl$_\downarrow$	ARC$_e$ acc	ARC$_c$ acc$_n$	Hella. acc$_n$	Lamb. acc	PIQA acc	Wino. acc	Avg.
2.7B parameters with 100B training tokens, L=32, d=2,560										
Xfmr++	N/A	10.7	15.2	59.8	27.5	54.2	52.3	72.7	**56.2**	53.8
Mamba	$64Ld$	13.6	15.9	60.7	29.8	53.9	46.4	72.8	53.9	52.9
RetNet	$512 \times Ld$	11.9	15.8	59.6	28.1	54.0	49.6	72.3	53.8	52.9
GLA	$256 \times Ld$	12.4	15.5	59.2	29.9	54.0	50.4	71.7	55.7	53.5
HGRN2	$128 \times Ld$	**8.8**	**14.6**	60.8	30.3	**58.7**	**55.4**	73.0	54.2	**55.4**
GSA	$128 \times Ld$	9.8	14.8	**61.9**	**30.7**	57.0	52.7	**73.5**	56.0	55.3

Recall-intensive Task Performance

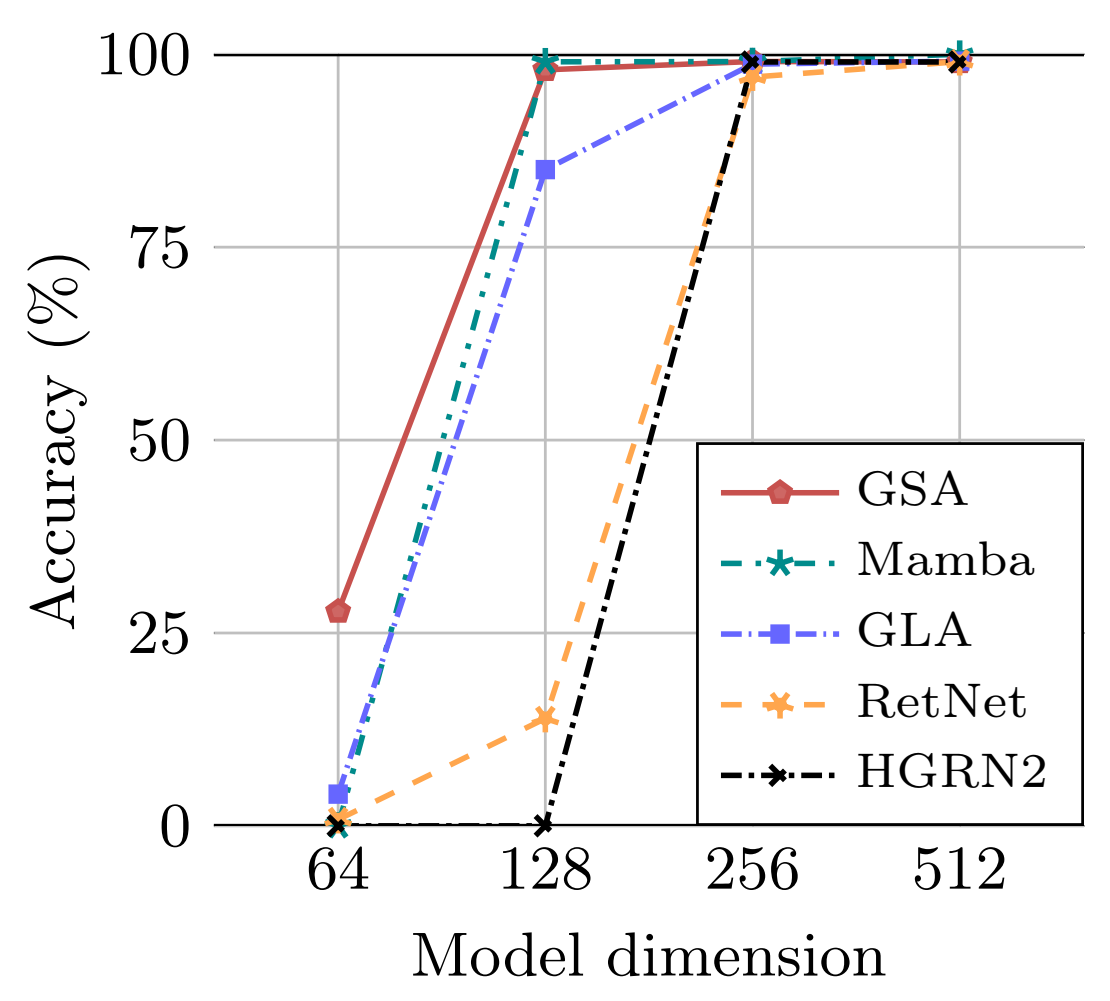


Figure 2. Results on the synthetic MQAR task. We adopt the most challenging settings in Arora et al. 2023. , utilizing a sequence length of 512 and 64 key-value pairs.

Figure 3. Results on the recall-intensive tasks.

	State size	FDA	SWDE	SQuAD	NQ	TriviaQA	Drop	Avg.
1.3B params / 100B tokens, L=24, d=2048								
Xfmr++	N/A	46.0	29.2	41.0	24.8	58.8	21.3	36.9
Mamba	$64 \times Ld$	13.9	25.4	33.2	18.5	53.5	**21.7**	27.7
RetNet	$512 \times Ld$	21.2	27.2	34.0	15.5	52.7	20.0	28.4
GLA	$256 \times Ld$	**26.7**	**30.6**	34.8	21.5	56.0	19.1	31.4
HGRN2	$128 \times Ld$	9.9	23.1	32.0	16.4	55.2	19.1	25.9
GSA	$128 \times Ld$	23.6	29.8	**36.0**	**23.2**	**57.0**	20.9	**31.8**
2.7B params / 100B tokens, L=32, d=2560								
Xfmr++	N/A	62.3	30.9	44.3	29.3	61.8	21.4	41.7
Mamba	$64 \times Ld$	21.5	26.7	34.2	21.2	57.0	**22.2**	30.5
RetNet	$512 \times Ld$	24.1	26.1	36.4	20.4	57.3	21.8	31.0
GLA	$256 \times Ld$	30.3	**35.5**	36.8	23.3	58.2	21.8	34.3
HGRN2	$128 \times Ld$	15.0	29.9	35.1	17.0	59.8	20.0	29.5
GSA	$128 \times Ld$	**39.1**	33.5	**39.0**	**26.9**	**60.8**	19.9	**36.5**

Finetuning Pretrained Transformers to RNNs

Table 3. Performance comparison across various 7B models.

	Size	Tokens	ARC$_e$	ARC$_c$	Hella.	PIQA	Wino.	NQ	TriviaQA	BBH	MMLU	Avg.
Shot(s)			0	0	0	0	0	5	5	3	5	
Models trained from scratch (for reference)												
RWKV6	7B	1.4T	73.6	44.0	75.2	78.4	68.5	20.9	59.5	23.4	43.9	54.1
Mistral♣	7B	?	80.8	54.0	81.1	80.6	74.0	29.7	70.3	56.5	62.4	65.5
Models finetuned from Mistral 7B												
SUPRA	7B	+20B	74.6	42.3	74.8	**80.1**	67.4	-	-	-	28.0	-
RetNet†	7B	+20B	73.3	39.9	72.9	77.8	66.1	16.2	43.0	8.7	26.1	47.1
GLA†	7B	+20B	74.6	**44.0**	75.9	79.2	69.5	22.2	57.8	20.8	28.4	52.5
GSA†	7B	+20B	**75.9**	43.9	**76.5**	78.7	**70.1**	**23.4**	**60.7**	**23.5**	**32.4**	**53.9**

NeurIPS 2024, Vancouver, Canada