Gated Slot Attention for Efficient Linear-Time Sequence Modeling

Yu Zhang

October 30, 2024

Institute of Artificial Intelligence, School of Computer Science and Technology,
Soochow University, Suzhou, China
yzhang.cs@outlook.com

Outline

1. Background
2. Gated Linear Attention
3. Gated Slot Attention
4. Experiments
5. Flash Linear Attention
6. Contributions

Background

Transformers

(Decoder-only) Transformers have been the de facto standard for training large language models (LLMs).

- GPT: [OpenAI et al., 2024]
- Llama: [Touvron et al., 2023]
- Qwen: [Yang et al., 2024a]

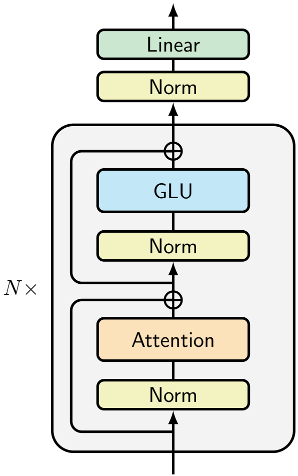


Figure 1: The backbone of Llama-like transformers (Xfmr++).

Transformers: Standard Attention

Standard Attention (SA) [Vaswani et al., 2017]

- Training: given $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T]^\top \in \mathbb{R}^{T \times d}$

$$\mathbf{O} = \mathrm{softmax}((\mathbf{Q}\mathbf{K}^\top) \odot \mathbf{M})\mathbf{V}, \tag{1}$$

  fully parallelizable, but needs $O(T^2 d)$ complexity

- Inference:

$$\mathbf{o}_t = \mathbf{V}_t^\top \mathrm{softmax}(\mathbf{K}_t \boldsymbol{q}_t), \tag{2}$$

  tokens are processed one by one, requiring $O(Td)$ memory to store the history

Linear Attention (LA) [Katharopoulos et al., 2020] emerged as a linear-time alternative to SA

- LA removes $\mathrm{softmax}$ in SA, and utilize the associative property to reduce FLOPs

$$\mathbf{O} = (\mathbf{Q}\mathbf{K}^\top)\mathbf{V} = \mathbf{Q}(\mathbf{K}^\top\mathbf{V}), \tag{3}$$

$\mathbf{K}^\top\mathbf{V}$ first in $O(Td^2)$, fully parallelizable as well

- Admits recurrent computation during inference

$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_{t-1} + \boldsymbol{k}_t \otimes \boldsymbol{v}_t \in \mathbb{R}^{d\times d} \\ \mathbf{o}_t &= \mathbf{S}_t^T \mathbf{q}_t \end{aligned} \tag{4}$$

only $O(d^2)$ matrix-valued memory to store the history

Transformers: Summarization

	f	Training parallelization		Inference cost	
SA	softmax	☺	$O(T^2d)$	☹	$O(Td)$
LA	-	☺	$O(Td^2)$	☺	$O(d^2)$

Key question: How to linearize SA?

- Generally, $\mathbf{K}, \mathbf{V}$ can be viewed as neural key-value memories $\widetilde{\mathbf{K}}_t, \widetilde{\mathbf{V}}_t \in \mathbb{R}^{m \times d}$ [Sukhbaatar et al., 2015]

$$\boldsymbol{o}_t = \widetilde{\mathbf{V}}_t^\top f(\widetilde{\mathbf{K}}_t \boldsymbol{q}_t). \tag{5}$$

- Transformers are equipped with unbounded number of memory slots, i.e., $m = t$ for step t
- simply fix the number of memory slots to a constant size $m \ll T$

Transformers as KV Memories: Sliding Window Attention

- **First-in-first-out**: the oldest one is popped out once a new key is introduced into the full buffer

$$\widetilde{\mathbf{K}}_t = \{\boldsymbol{k}_{t-m}, \ldots, \boldsymbol{k}_t\}$$

- Information outside the window strategy is discarded, necessitating a large window size, e.g., 4K tokens in Mistral

Gated Linear Attention

Attention with bounded-memory control

ABC [Peng et al., 2022]

- When $T > m$, saving information from multiple tokens into one slot is inevitable.
- ABC defines control vector $\boldsymbol{\phi}_t \in \mathbb{R}^m$ for memory read/writing:

$$\widetilde{\mathbf{K}} = \sum_{i=1}^{T} \boldsymbol{\phi}_i \otimes \boldsymbol{k}_i \in \mathbb{R}^{m \times d}, \quad \widetilde{\mathbf{V}} = \sum_{i=1}^{T} \boldsymbol{\phi}_i \otimes \boldsymbol{v}_i \in \mathbb{R}^{m \times d}, \tag{6}$$

- ABC admits recurrent computation, involving two-pass LA

$$\widetilde{\mathbf{K}}_t = \widetilde{\mathbf{K}}_{t-1} + \boldsymbol{\phi}_t \otimes \boldsymbol{k}_t \in \mathbb{R}^{m \times d}, \quad \widetilde{\mathbf{V}}_t = \widetilde{\mathbf{V}}_{t-1} + \boldsymbol{\phi}_t \otimes \boldsymbol{v}_t \in \mathbb{R}^{m \times d} \tag{7}$$

$$\mathbf{o}_t = \widetilde{\mathbf{V}}^T \operatorname{softmax}(\widetilde{\mathbf{K}}_t^T \mathbf{q}_t) \in \mathbb{R}^d. \tag{8}$$

- Naive implementations are expensive

Attention with bounded-memory control

	f	Forget gate	Training parallelization	Inference cost
SA	softmax	-	☺ $O(T^2d)$	☹ $O(Td)$
LA	-	☹	☺ $O(Td^2)$	☺ $O(d^2)$
ABC	softmax	☹	😑 $O(Td^2)$	😑 $O(d^2)$

Data-independent gating

RetNet [Sun et al., 2023]

- Upon vanilla LA, RetNet introduces forget gate to control the decay rate

$$\mathbf{S}_t = \gamma \mathbf{S}_{t-1} + \boldsymbol{k}_t \otimes \boldsymbol{v}_t \in \mathbb{R}^{d \times d},$$

  $\gamma \in (0, 1)$ is a scalar <u>data-independent</u> decaying factor

- The decay rate is fixed across time steps, regardless of the input tokens

Data-dependent gating

GLA [Yang et al., 2024b]

- It is shown that data-dependent decay is crucial for RNNs to selectively retain and forget infos [Gu and Dao, 2023]
- GLA introduces forget gate depending on the input

$$\mathbf{S}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t) \cdot \mathbf{S}_{t-1} + \boldsymbol{k}_t \otimes \boldsymbol{v}_t \in \mathbb{R}^{d \times d}, \quad \boldsymbol{o}_t = \mathbf{S}_t^T \boldsymbol{q}_t \in \mathbb{R}^d.$$

Summarization

	f	Forget gate		Training parallelization		Inference cost	
LA	-	☹		☺	$O(Td^2)$	☺	$O(d^2)$
ABC	softmax	☹		😑	$O(Td^2)$	😑	$O(d^2)$
RetNet	-	😑	γ	☺	$O(Td^2)$	☺	$O(d^2)$
Mamba2	-	☺	γ_t	☺	$O(Td^2)$	☺	$O(d^2)$
GLA	-	☺	$\boldsymbol{\alpha}_t$	☺	$O(Td^2)$	☺	$O(d^2)$

Gated Slot Attention

GSA

GSA improves ABC using a selective gating mechanism.

- For each memory slot, the update rule is a simple gated RNN with a scalar gating value

$$\begin{aligned} \widetilde{\mathbf{K}}_t &= \mathrm{Diag}(\boldsymbol{\alpha}_t) \cdot \widetilde{\mathbf{K}}_{t-1} + (1 - \boldsymbol{\alpha}_t) \otimes \boldsymbol{k}_t \in \mathbb{R}^{m \times d} \\ \widetilde{\mathbf{V}}_t &= \mathrm{Diag}(\boldsymbol{\alpha}_t) \cdot \widetilde{\mathbf{V}}_{t-1} + (1 - \boldsymbol{\alpha}_t) \otimes \boldsymbol{v}_t \in \mathbb{R}^{m \times d} \\ \mathbf{o}_t &= \widetilde{\mathbf{V}}^T \, \mathrm{softmax}(\widetilde{\mathbf{K}}_t^T \mathbf{q}_t) \in \mathbb{R}^d \end{aligned} \tag{9}$$

GSA as two-pass GLA

$$
\begin{aligned}
\{\boldsymbol{o}'_i\}_{i=1}^{T} &= \mathrm{GLA}\left(\{\boldsymbol{q}_i, \boldsymbol{k}_i, 1-\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_i, \mathbf{1}\}_{i=1}^{T}\right) \\
\{\boldsymbol{o}_i\}_{i=1}^{T} &= \mathrm{GLA}\left(\{\mathrm{softmax}(\boldsymbol{o}'_i), 1-\boldsymbol{\alpha}_i, \boldsymbol{v}_i, \mathbf{1}, \boldsymbol{\alpha}_i\}_{i=1}^{T}\right)
\end{aligned} \tag{10}
$$

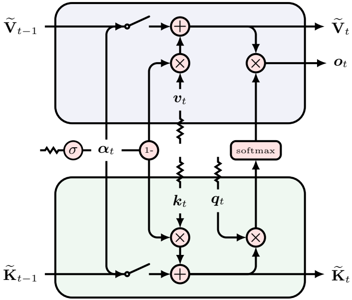


Figure 2: The recurrent representation of GSA. ⟿ means taking $\boldsymbol{x}_t$ as input.

Remarks

- Queries in GSA is the output of the first GLA pass and thereby is aware of the entire historical information.
- GSA preserves the $\mathrm{softmax}$ operator, more suitable in the setting of "fine-tuning pretrained Transformers into RNNs" [Kasai et al., 2021].
- GSA only needs half the recurrent state size of GLA and a quarter the recurrent size of RetNet, while having better performance.

Summarization

	f	Forget gate		Training parallelization		Inference cost	
LA	-	☹		☺	$O(Td^2)$	☺	$O(d^2)$
ABC	softmax	☹		😑	$O(Td^2)$	😑	$O(d^2)$
RetNet	-	😑	γ	☺	$O(Td^2)$	☺	$O(d^2)$
Mamba2	-	☺	γ_t	☺	$O(Td^2)$	☺	$O(d^2)$
GLA	-	☺	$\boldsymbol{\alpha}_t$	☺	$O(Td^2)$	☺	$O(d^2)$
GSA	softmax	☺	$\boldsymbol{\alpha}_t$	☺	$O(Td^2)$	☺	$O(d^2)$

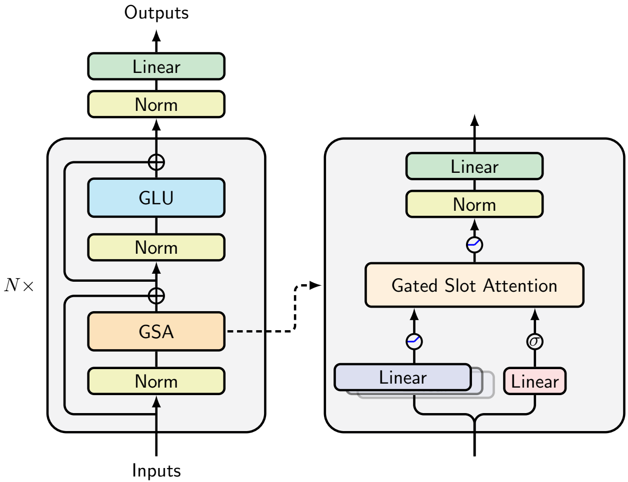


Figure 3: The backbone of our proposed GSA models. The main architecture follows the settings of Llama that stacks token-mixing layer (our GSA) and channel-wise Swish GLU alternatively.

Related Works

HGRN [Qin et al., 2023]

- Like LRU [Orvieto et al., 2023], ll non-linearities are removed, enabling HGRN to be parallelized in $O(T \log T)$ time.
- HGRN introduces data-dependent gating as well as tied input gates following GRU.

$$\begin{aligned} \boldsymbol{h}_t &= \boldsymbol{f}_t \odot \boldsymbol{h}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{c}_t \in \mathbb{R}^d \\ \boldsymbol{i}_t &= \mathbf{1} - \boldsymbol{f}_t \in \mathbb{R}^d \end{aligned} \tag{11}$$

Mamba [Gu and Dao, 2023]

- Data-dependent gating with (1-d) state expansion.

$$h_t = \bar{\mathbf{A}}_t h_{t-1} + \bar{\mathbf{B}}_t x_t \in \mathbb{R}^N \tag{12}$$

HGRN2 [Qin et al., 2024]

- 2-d state expansion as well as tied input gates upon GLA.

$$\mathbf{S}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t) \cdot \mathbf{S}_{t-1} + (\mathbf{1} - \boldsymbol{\alpha}_t) \otimes \boldsymbol{v}_t \in \mathbb{R}^{d \times d}. \tag{13}$$

DeltaNet [Schlag et al., 2021, Yang et al., 2024c]

- Erase infos before writing new ones.

$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_{t-1} - \boldsymbol{k}_t \otimes \boldsymbol{v}_t^{old} + \boldsymbol{k}_t \otimes \boldsymbol{v}_t^{new} \\ &= \mathbf{S}_{t-1} - \beta_t \boldsymbol{k}_t \otimes (\mathbf{S}_{t-1}^\top \boldsymbol{k}_t) + \beta_t \boldsymbol{k}_t \otimes \boldsymbol{v}_t \\ &= (\mathbf{1} - \beta_t \boldsymbol{k}_t \otimes \boldsymbol{k}_t) \mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \otimes \boldsymbol{v}_t. \end{aligned} \tag{14}$$

Related Works

TTT [Sun et al., 2024]

- Formulate the updating rule as an explicit step of optimization

$$\mathbf{W}_t = \mathbf{W}_{t-1} - \eta \nabla l(\mathbf{W}_{t-1}; \boldsymbol{x}_t) \tag{15}$$

- To make the gradient computation tractable, l is simply $\|f(\tilde{\boldsymbol{x}}_t; \boldsymbol{W}) - \boldsymbol{x}_t\|^2$.
- When f is a sophisticated neural network, TTT can be hard to parallelize.
- When f is a simple linear layer, TTT degenerates to LA.

Experiments

Baselines

- Xfmr++ [Touvron et al., 2023]: Llama-like architectures that enhance the vanilla Transformer by using Rotary position embeddings and GLU ([Shazeer, 2020]);
- Mamba [Gu and Dao, 2023]: State space models with data-dependent gating;
- RetNet [Sun et al., 2023]: Linear attention with non-learnable, data-independent decay;
- GLA [Yang et al., 2024b]: Linear attention with elementwise, data-dependent decay.

Language Modeling

Table 1: The zero-shot results of 1.3B and 2.7B models evaluated by `lm-evaluation-harness`. The rightmost column shows the average results of all (normalized) acc scores.

	State size	Lamb. ppl↓	Wiki. ppl↓	ARC_e acc	ARC_c acc_n	Hella. acc_n	Lamb. acc	PIQA acc	Wino. acc	Avg.
1.3B parameters with 100B training tokens, L=24, d=2048										
Xfmr++	N/A	15.3	17.1	54.1	27.1	49.3	47.0	70.3	54.9	50.5
Mamba	$64 \times Ld$	16.5	18.2	57.3	26.6	48.1	43.4	69.5	53.7	49.8
RetNet	$512 \times Ld$	15.4	17.3	57.4	27.9	50.3	44.6	71.7	51.8	50.6
GLA	$256 \times Ld$	15.4	17.6	55.4	27.7	49.0	46.4	69.9	54.0	50.4
GSA (ours)	$128 \times Ld$	12.6	16.7	58.1	28.2	51.0	47.4	72.0	53.4	**51.7**
2.7B parameters with 100B training tokens, L=32, d=2560										
Xfmr++	N/A	10.7	15.2	59.8	27.5	54.2	52.3	72.7	56.2	53.8
Mamba	$64 \times Ld$	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
GSA (ours)	$128 \times Ld$	9.8	14.8	61.9	30.7	57.0	52.7	73.5	56.0	**55.3**

Recall-intensive tasks

Table 2: Results on 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

Recall-intensive tasks

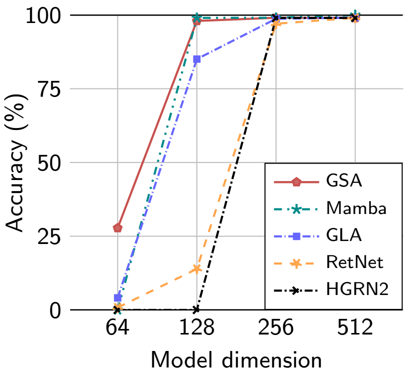


Figure 4: Results on the synthetic MQAR task. We adopt the most challenging settings in [Arora et al., 2023], utilizing a sequence length of 512 and 64 key-value pairs. Xfmr++ with standard attention achieves near-perfect results in this settings and is thus omitted for brevity.

Ablations

Table 3: Ablation study results for 340M models trained on 10B Slimpajama tokens.

	PPL (↓)
GSA w/ 64 slots	13.51
Ablations on gating mechanism	
w/o decay (i.e., ABC)	16.94
w/ data-independent decay	15.83
Ablations on non-linearity	
$-\mathrm{softmax}$	14.03
$-\mathrm{softmax} + \mathrm{Swish}$	13.71
$-\mathrm{softmax} + \mathrm{ReLU}$	13.69
$-\mathrm{softmax} + \mathrm{ReLU}^2$	13.95
Ablations on slot size	
w/ 32 slots	13.74
w/ 128 slots	**13.46**

Training Efficiency

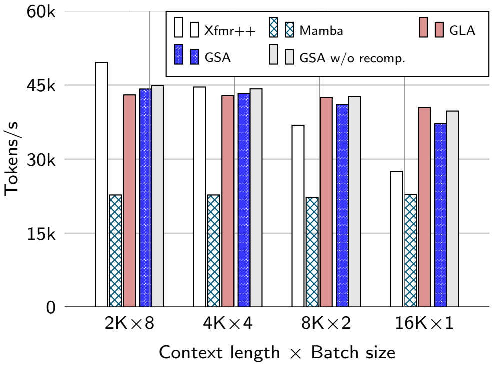


Figure 5: Training throughput of various 1.3B models on a single H800 GPU, with a fixed batch size containing 16K tokens. "GSA w/o recomp." indicates the use of the GSA kernel without hidden state recomputation during the backward pass.

Inference Speed

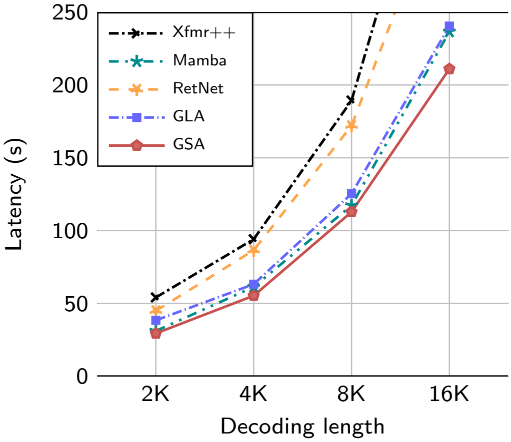


Figure 6: Inference speed of different models with 1.3B parameters.

Continual Pretraining

Table 4: Performance comparison across various 7B models. ♣ denotes models using softmax-attention. † denotes our results.

	Size	Tokens	ARC_e	ARC_c	Hella.	PIQA	Wino.	NQ	TriviaQA	BBH	MMLU	Avg.
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
Mamba	7B	1.2T	77.6	46.8	77.8	81.0	72.3	25.4	66.2	21.5	33.2	55.7
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**
SUPRA	7B	+100B	**76.0**	45.7	77.1	**79.9**	70.3	24.7	60.4	19.8	34.1	54.2
GSA†	7B	+100B	**76.0**	**46.9**	**77.9**	78.9	**72.6**	**26.9**	**65.8**	**29.3**	**38.1**	**56.9**

Flash Linear Attention

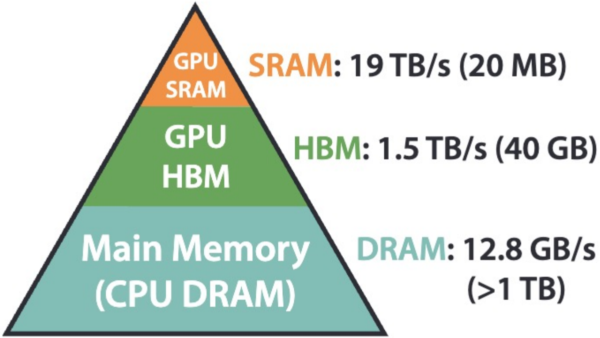


Figure 7: Memory Hierarchy with Bandwidth & Memory Size.

Hardware-aware Considerations

- IO-aware: reduce IO transmission between SRAM and HBM.
- Matrix Multiplication with tensor-cores can be $16\times$ faster than CUDA cores.
    - Flash Attention ☺
    - Mamba ☹

- Recurrent form: inefficient during training, preventing the full utilization of modern GPU parallelism over sequence lengths

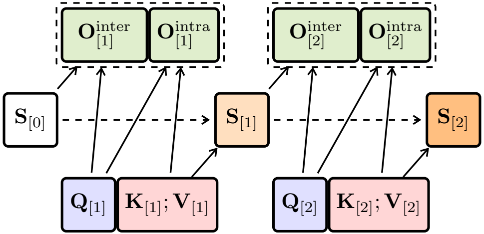
$$\mathbf{S}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t)\mathbf{S}_{t-1} + \boldsymbol{k}_t \otimes \boldsymbol{v}_t \tag{16}$$

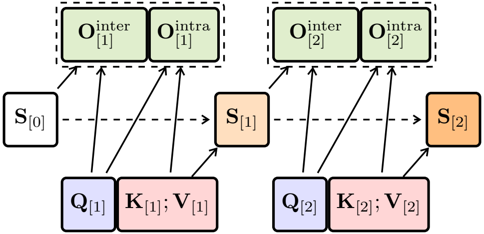
- Parallel form: can be parallelized in similar vein as in flash attention [Dao et al., 2022], but still adheres to the quadratic complexity

$$\mathbf{O} = ((\mathbf{Q}\mathbf{K}^\top) \odot \mathbf{M})\mathbf{V}. \tag{17}$$

FLA: chunkwise-form parallelism

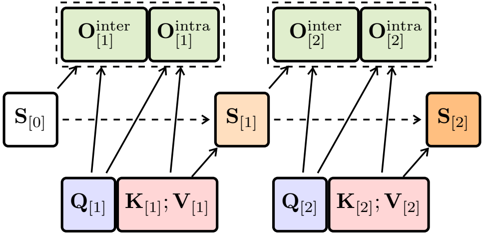
Chunk $\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}$ to $\mathbf{Q}_{[i]}, \mathbf{K}_{[i]}, \mathbf{V}_{[i]}, \mathbf{O}_{[i]} \in \mathbb{R}^{C\times d}$





$$\mathbf{S}_{[i]} = \mathbf{S}_{[i-1]} + \underbrace{\sum_{j=iC+1}^{(i+1)C} \boldsymbol{k}_j \otimes \boldsymbol{v}_j}_{\mathbf{K}_{[i]}^{\top}\mathbf{V}_{[i]}} \in \mathbb{R}^{d\times d} \tag{18}$$

FLA: chunkwise-form parallelism



$$\mathbf{O}_{[i]} = \underbrace{\mathbf{Q}_{[i]}\mathbf{S}_{[i-1]}}_{\mathbf{O}^{\text{inter}}_{[i]}} + \underbrace{((\mathbf{Q}_{[i]}\mathbf{K}^{\top}_{[i]}) \odot \mathbf{M})\mathbf{V}_{[i]}}_{\mathbf{O}^{\text{intra}}_{[i]}} \in \mathbb{R}^{C\times d} \tag{19}$$

Figure 8: FLA [Yang and Zhang, 2024]: A Triton-Based Library for Hardware-Efficient Implementations of Linear Attention Mechanism

FLA

https://github.com/sustcsonglin/flash-linear-attention

Contributions

Contributions

- We introduces Gated Slot Attention (GSA), a new attention variant that admits linear complexity.
- We incorporate a data-dependent gating mechanism to effectively update the memories, which is crucial for language modeling performance.
- We verify the effectiveness of GSA by training it from scratch on models with 1.3B and 2.7B parameters.
- We also show that GSA benefits from maintaining the $\mathrm{softmax}$ formulation, making it more amenable to linearizing the well-trained SA-based LLMs.

Q & A

Arora, S., Eyuboglu, S., Timalsina, A., Johnson, I., Poli, M., Zou, J., Rudra, A., and Ré, C. (2023).
Zoology: Measuring and improving recall in efficient language models.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. (2022).
Flashattention: Fast and memory-efficient exact attention with io-awareness.
In Advances in NIPS, pages 16344–16359.

Gu, A. and Dao, T. (2023).
Mamba: Linear-time sequence modeling with selective state spaces.

Kasai, J., Peng, H., Zhang, Y., Yogatama, D., Ilharco, G., Pappas, N., Mao, Y., Chen, W., and Smith, N. A. (2021).
Finetuning pretrained transformers into RNNs.
In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, Proceedings of EMNLP, pages 10630–10643.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020).
Transformers are RNNs: Fast autoregressive transformers with linear attention.
In III, H. D. and Singh, A., editors, Proceedings of ICML, pages 5156–5165. PMLR.

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, J. H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V. H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder,

N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M. B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. (2024).
Gpt-4 technical report.

Orvieto, A., Smith, S. L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., and De, S. (2023).
Resurrecting recurrent neural networks for long sequences.

Peng, H., Kasai, J., Pappas, N., Yogatama, D., Wu, Z., Kong, L., Schwartz, R., and Smith, N. A. (2022).
ABC: Attention with bounded-memory control.
In Proceedings of ACL, pages 7469–7483.

Qin, Z., Yang, S., Sun, W., Shen, X., Li, D., Sun, W., and Zhong, Y. (2024).
Hgrn2: Gated linear rnns with state expansion.

Qin, Z., Yang, S., and Zhong, Y. (2023).
Hierarchically gated recurrent neural network for sequence modeling.
In Advances in NIPS.

Schlag, I., Irie, K., and Schmidhuber, J. (2021).
Linear transformers are secretly fast weight programmers.
In Meila, M. and Zhang, T., editors, Proceedings of ICML, pages 9355–9366. PMLR.

Shazeer, N. (2020).
Glu variants improve transformer.

Sukhbaatar, S., szlam, a., Weston, J., and Fergus, R. (2015).
End-to-end memory networks.
In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, Advances in NIPS. Curran Associates, Inc.

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. (2023).
Retentive network: A successor to transformer for large language models.

- Sun, Y., Li, X., Dalal, K., Xu, J., Vikram, A., Zhang, G., Dubois, Y., Chen, X., Wang, X., Koyejo, S., Hashimoto, T., and Guestrin, C. (2024).
  Learning to (learn at test time): Rnns with expressive hidden states.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023).

Llama 2: Open foundation and fine-tuned chat models.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017).
Attention is all you need.
In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, Advances in NIPS. Curran Associates, Inc.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., Yang, J., Xu, J., Zhou, J., Bai, J., He, J., Lin, J., Dang, K., Lu, K., Chen, K., Yang, K., Li, M., Xue, M., Ni, N., Zhang, P., Wang, P., Peng, R., Men, R., Gao, R., Lin, R., Wang, S., Bai, S., Tan, S., Zhu, T., Li, T., Liu, T., Ge, W., Deng, X., Zhou, X., Ren, X., Zhang, X., Wei, X., Ren, X., Liu, X., Fan, Y., Yao, Y., Zhang, Y., Wan, Y., Chu, Y., Liu, Y., Cui, Z., Zhang, Z., Guo, Z., and Fan, Z. (2024a).

Qwen2 technical report.

Yang, S., Wang, B., Shen, Y., Panda, R., and Kim, Y. (2024b).
Gated linear attention transformers with hardware-efficient training.
In Proceedings of ICML. PMLR.

Yang, S., Wang, B., Zhang, Y., Shen, Y., and Kim, Y. (2024c).
Parallelizing linear transformers with the delta rule over sequence length.
ArXiv, abs/2406.06484.

Yang, S. and Zhang, Y. (2024).
FLA: A Triton-Based Library for Hardware-Efficient Implementations of Linear Attention Mechanism.