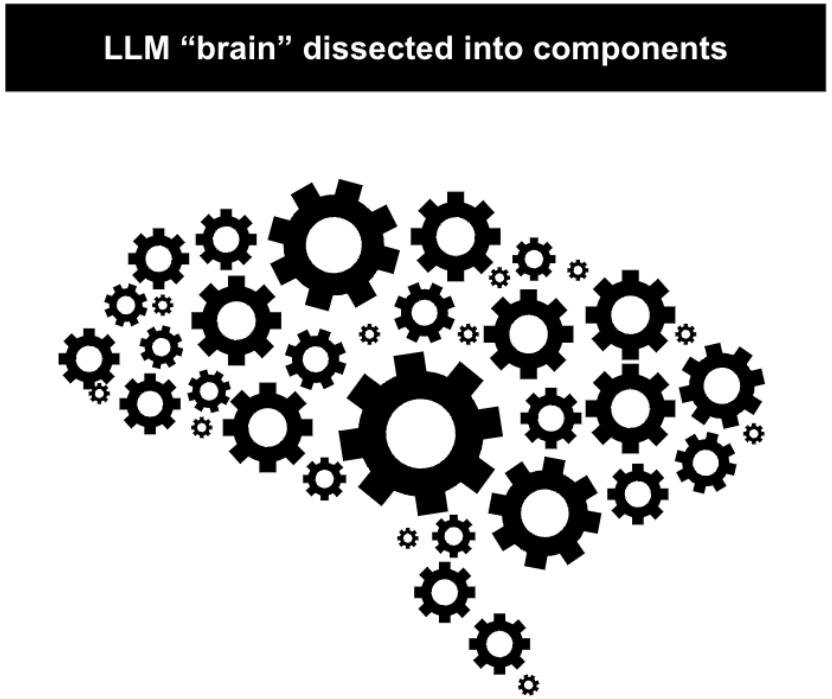
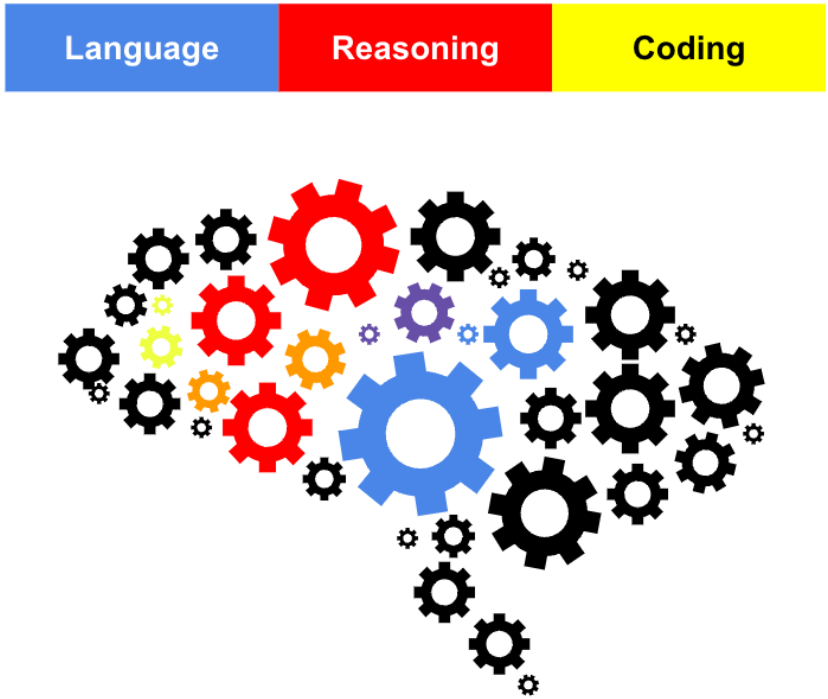
TRANSFORMER-SQUARED: SELF-ADAPTIVE LLMS

Sakana AI

TRANSFORMER-SQUARED: SELF-ADAPTIVE LLMS

- 认为预训练后的大模型已拥有充分的能力，重点在于如何表达
- 认为现有的类LoRA的高效参数微调方法均为表征模拟
- 提出了一种动态调整大模型参数权重的微调方法





TRANSFORMER-SQUARED: SELF-ADAPTIVE LLMS

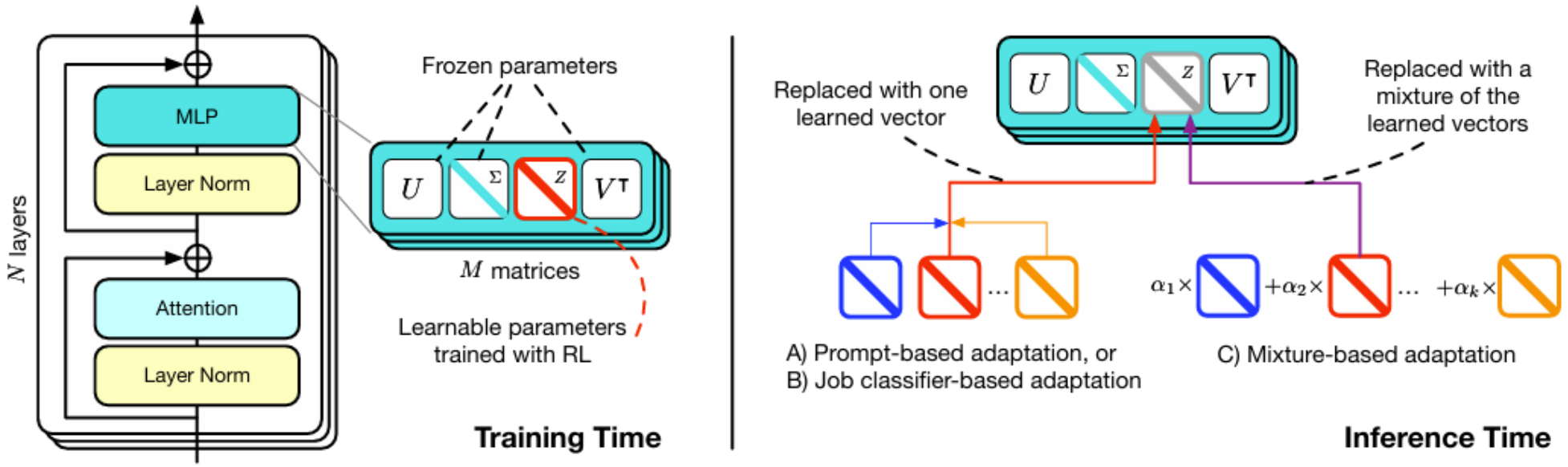


Figure 2: **Method overview.** Left) At training time, we employ SVF and RL to learn the "expert" vectors z's that scale the singular values of the weight matrices. Right) At inference time, we propose three distinct methods to adaptively select/combine the learned expert vectors.

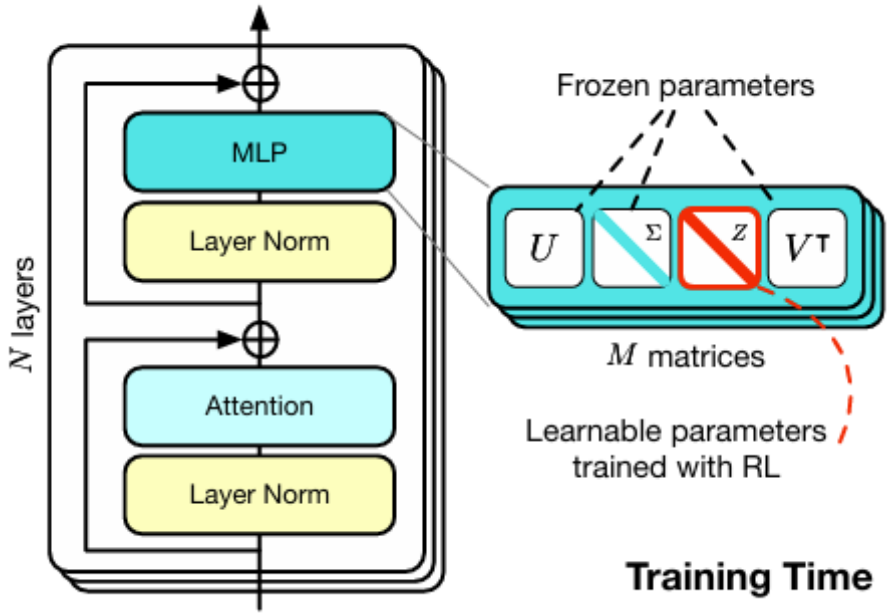
- 奇异值微调（SVF）
- 权重自适应

TRANSFORMER-SQUARED: SELF-ADAPTIVE LLMS

奇异值微调（SVF）

学习一个简单的向量 z
改变奇异值 $\Sigma' = \Sigma \otimes diag(z)$
权重矩阵更改为$W' = U\Sigma' V^T$

$$J(\theta_z) = \mathbb{E}\left[\log\left(\pi_{\theta_{W'}}(\hat{y}_i \mid x_i)\right) r(\hat{y}_i, y_i)\right] - \lambda D_{\mathrm{KL}}(\pi_{\theta_{W'}} \| \pi_{\theta_W})$$



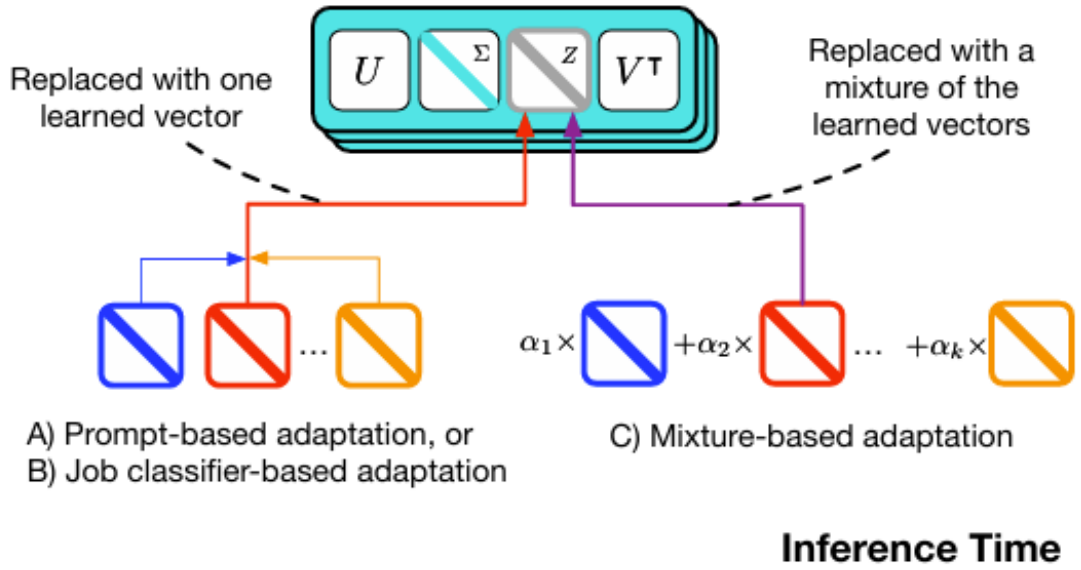
For example, suppose SVD decomposes a weight matrix into five components [A, B, C, D, E]. For a math task, the learned z-vector might be [1, 0.8, 0, 0.3, 0.5], meaning that component A is critical for math while component C hardly affects its performance. For a language understanding task, the z-vector could be [0.1, 0.3, 1, 0.7, 0.5], highlighting that component C is essential for this task despite being less useful for math.

TRANSFORMER-SQUARED: SELF-ADAPTIVE LLMS

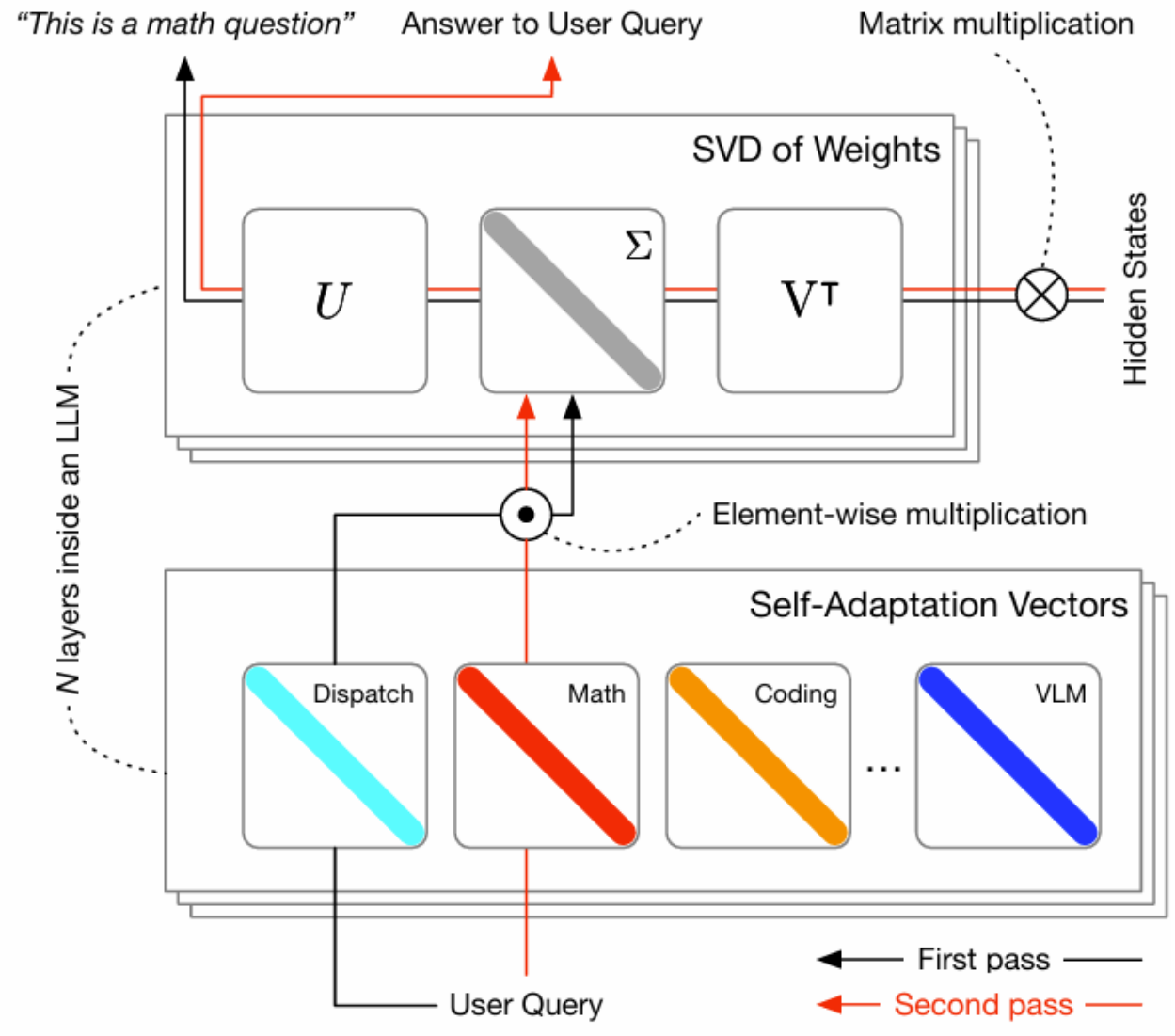
自适应性

- 基于prompt的适应方法
- 基于分类器的适应方法
- 融合适应方法（少样本适应）

$$z' = \sum_{k=1}^{K} \alpha_k z_k$$



TRANSFORMER-SQUARED: SELF-ADAPTIVE LLMS



TRANSFORMER-SQUARED: SELF-ADAPTIVE LLMS

北京大学 PEKING UNIVERSITY

Table 1: **Fine-tuning results.** LLM performance on the test splits of math, coding and reasoning. Normalized scores are in the parentheses.

Method	GSM8K	MBPP-Pro	ARC-Easy
LLAMA3-8B-INSTRUCT	75.89 (1.00)	64.65 (1.00)	88.59 (1.00)
+ LoRA	77.18 (1.02)	**67.68** (**1.05**)	88.97 (1.00)
+ SVF (Ours)	**79.15** (**1.04**)	66.67 (1.03)	**89.56** (**1.01**)
MISTRAL-7B-INSTRUCT-V0.3	42.83 (1.00)	49.50 (1.00)	81.65 (1.00)
+ LoRA	44.66 (1.04)	51.52 (1.04)	81.19 (0.98)
+ SVF (Ours)	**49.74** (**1.16**)	**51.52** (**1.04**)	**85.14** (**1.04**)
LLAMA3-70B-INSTRUCT	85.29 (1.00)	**80.81** (**1.00**)	**89.10** (**1.00**)
+ LoRA	77.26 (0.91)	68.69 (0.85)	88.55 (0.99)
+ SVF (Ours)	**88.32** (**1.04**)	**80.81** (1.00)	88.47 (0.99)

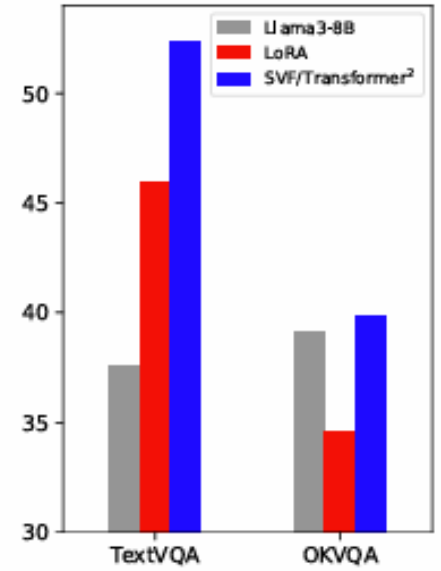


Figure 5: **Results for the VLM domain.**

TRANSFORMER-SQUARED: SELF-ADAPTIVE LLMS

北京大学 PEKING UNIVERSITY

Table 2: **Self-adaptation on unseen tasks.** Normalized scores are in the parentheses.

Method	MATH	Humaneval	ARC-Challenge
LLAMA3-8B-INSTRUCT 3	24.54 (1.00)	60.98 (1.00)	80.63 (1.00)
+ LoRA	24.12 (0.98)	52.44 (0.86)	81.06 (1.01)
+ Transformer^2 (Prompt)	25.22 (1.03)	61.59 (1.01)	81.74 (1.01)
+ Transformer^2 (Cls-expert)	25.18 (1.03)	62.80 (1.03)	81.37 (1.01)
+ Transformer^2 (Few-shot)	**25.47 (1.04)**	**62.99 (1.03)**	**82.61 (1.02)**
MISTRAL-7B-INSTRUCT-V0.3	13.02 (1.00)	43.29 (1.00)	71.76 (1.00)
+ LoRA	13.16 (1.01)	37.80 (0.87)	**75.77 (1.06)**
+ Transformer^2 (Prompt)	11.86 (0.91)	43.90 (1.01)	72.35 (1.01)
+ Transformer^2 (Cls-expert)	11.60 (0.89)	43.90 (1.01)	74.83 (1.04)
+ Transformer^2 (Few-shot)	**13.39 (1.03)**	**47.40 (1.09)**	75.47 (1.05)
LLAMA3-70B-INSTRUCT	**40.64 (1.00)**	78.66 (1.00)	87.63 (1.00)
+ LoRA	25.40 (0.62)	73.78 (0.94)	83.70 (0.96)
+ Transformer^2 (Prompt)	40.44 (1.00)	**79.88 (1.02)**	**88.48 (1.01)**

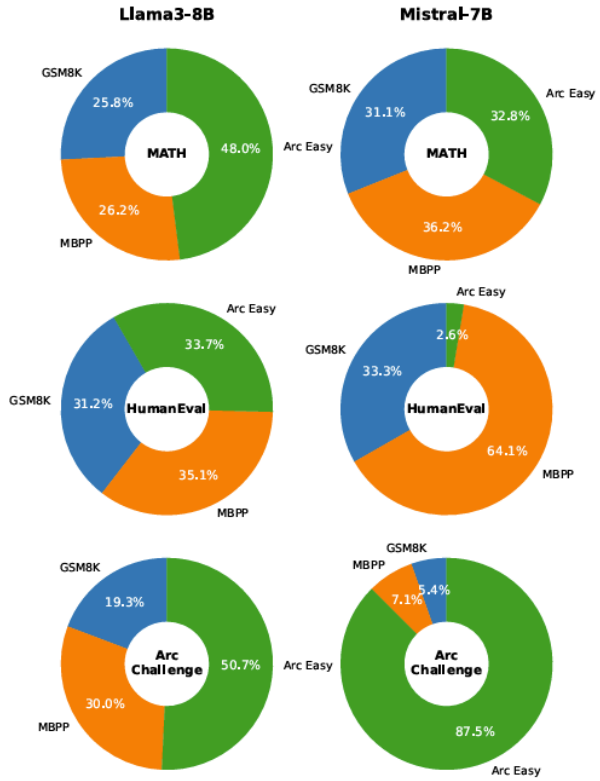


Figure 7: α_k **learned weights.**

TRANSFORMER-SQUARED: SELF-ADAPTIVE LLMS

北京大学 PEKING UNIVERSITY

Table 3: **Time cost of 2-pass inference in prompt adaptation strategy of Transformer^2 for the entire problem set.** 1st to 2nd pass inference time ratios are shown in parentheses.

Task	1st (s)	2nd (s)
MATH	42.64 (13%)	321.19
Humaneval	2.76 (19%)	14.28
ARC-Challenge	13.40 (47%)	28.51

Table 4: **Ablation studies.** We fine-tune LLAMA3-8B-INSTRUCT on the GSM8K training split with different settings and the results on the test split along with zero-shot transfer results on MATH.

#	Method	Objective Function	Module	#Params (↓)	GSM8K (↑)	MATH (↑)
0	LLAMA-3-8B-INSTRUCT				75.89 (1.00)	24.54 (1.00)
1	SVF	Policy gradient	MLP	0.39M	78.62 (1.04)	24.20 (0.99)
2	SVF	Policy gradient	attention	**0.16M**	76.19 (1.00)	24.20 (0.99)
3	SVF	Policy gradient	MLP + attention	0.58M	**79.23 (1.04)**	**25.04 (1.04)**
4	SVF	Next token pred	attention	**0.16M**	60.50 (0.80)	18.52 (0.75)
5	LoRA	Policy gradient	attention	6.82M	57.92 (0.76)	15.72 (0.64)
6	LoRA	Next token pred	attention	6.82M	77.18 (0.98)	24.12 (0.96)
7	LoRA	Next token pred	MLP + attention	35.13M	75.66 (0.96)	22.12 (0.91)

Titans: Learning to Memorize at Test Time

Google

Titans: Learning to Memorize at Test Time

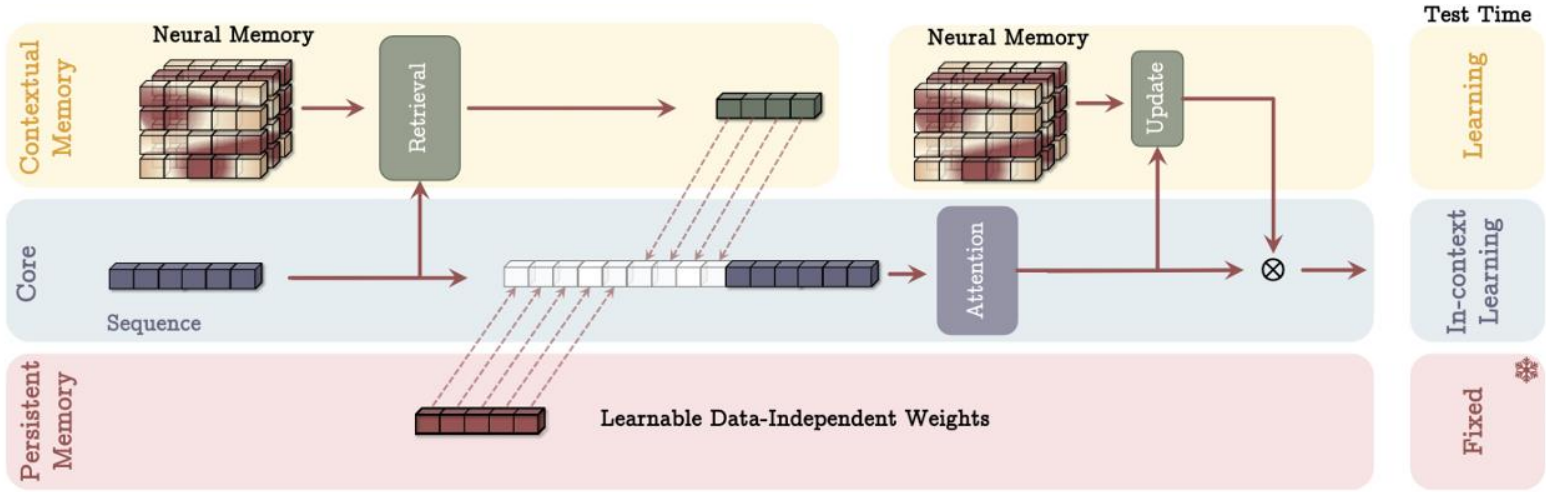


Figure 2: **Memory as a Context (MAC) Architecture.** This architecture includes three branches of (1) core, (2) contextual (long-term) memory, and (3) persistent memory. The core branch concatenates the *corresponding* long-term and persistent memories with the input sequence. Next, attention performs on the sequence and decides what part of the information should store in the long-term memory. At the test time, parameters corresponds to contextual memory are still learning, parameters corresponds to the core branch are responsible for in-context learning, and parameters of persistent memory are responsible to store the knowledge about tasks and so are fixed.

- Core
- Contextual Memory
- Persistent Memory

Titans: Learning to Memorize at Test Time

Long-term Memory

惊喜值：越偏离预期的行为越容易被记忆

$$\mathcal{M}_t = \mathcal{M}_{t-1} + S_t,$$
$$S_t = \eta_t \underbrace{S_{t-1}}_{\text{Past Surprise}} - \theta_t \underbrace{\nabla \ell\left(M_{t-1}; x_t\right)}_{\text{Momentary Surprise}}.$$

元模型：用于学习如何在训练阶段记忆k与v的映射

$$\ell(\mathcal{M}_{t-1}; x_t) = \|\mathcal{M}_{t-1}\left(\mathbf{k}_t\right) - \mathbf{v}_t\|_2^2$$

遗忘机制：进行显式遗忘

$$\mathcal{M}_t = (1 - \alpha_t)\mathcal{M}_{t-1} + S_t,$$
$$S_t = \eta_t S_{t-1} - \theta_t \nabla \ell\left(M_{t-1}; x_t\right),$$

Titans: Learning to Memorize at Test Time

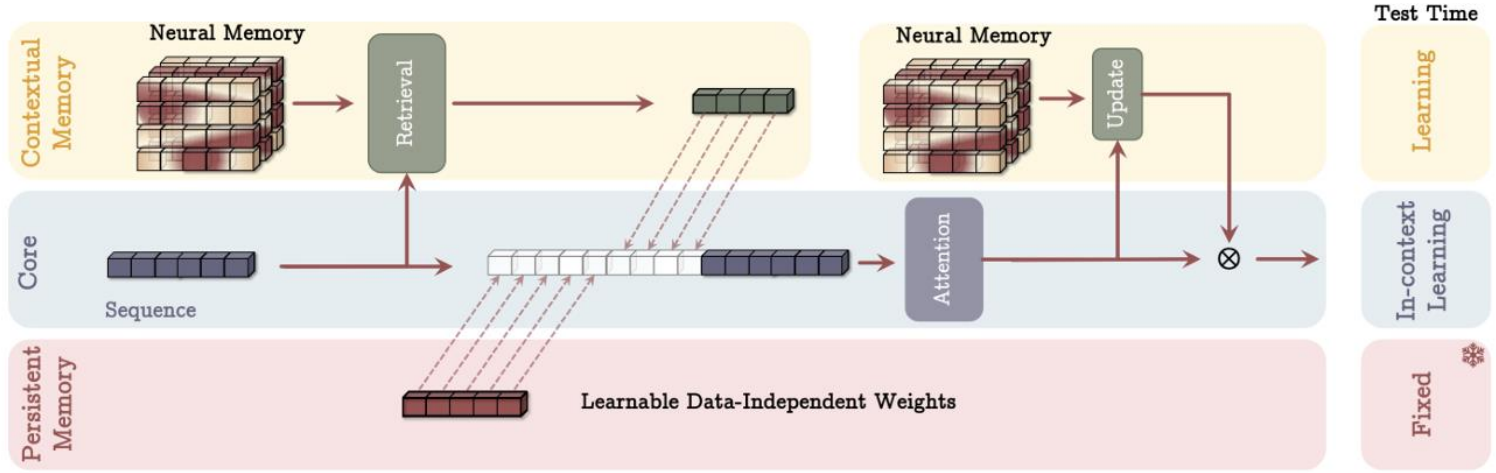
Persistent Memory（Context-Independent）

$$x_{\text{new}} = \begin{bmatrix} p_1 & p_2 & \cdots & p_{N_p} \end{bmatrix} \| \, x,$$

Motivation

- Memory Perspective
- Feedforward Network Perspective
- Technical Perspective

Titans: Learning to Memorize at Test Time



Memory as a Context

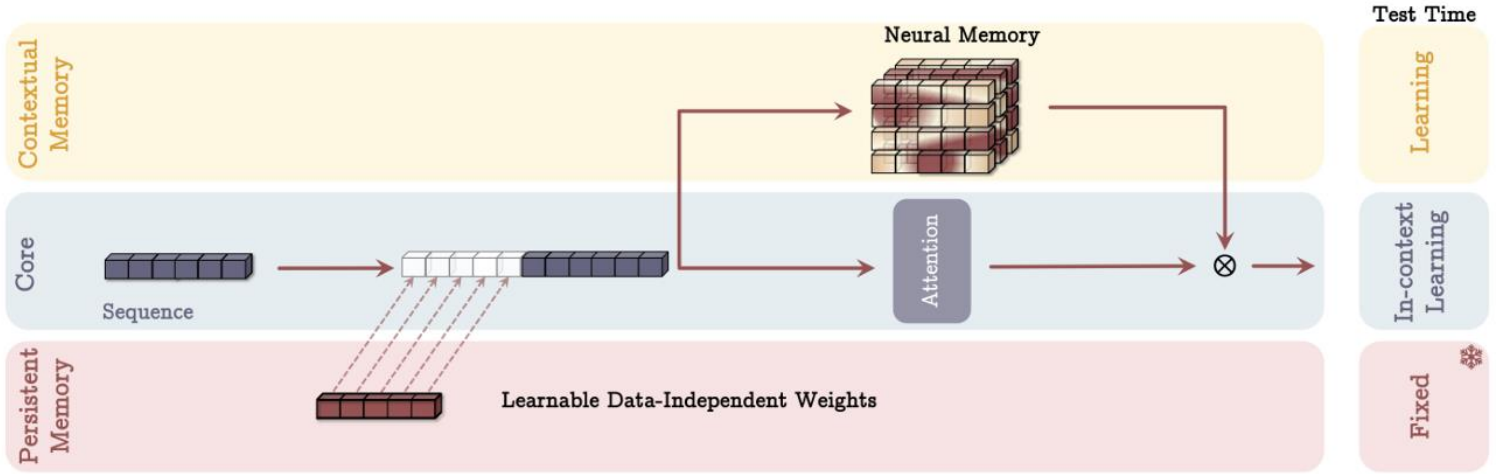
$$h_t = \mathcal{M}^*_{t-1}(\mathbf{q}_t),$$

$$\tilde{\mathrm{S}}^{(t)} = \begin{bmatrix} p_1 & p_2 & \dots & p_{N_p} \end{bmatrix} \,||\, h_t \,||\, \mathrm{S}^{(t)},$$

$$y_t = \mathtt{Attn}\left(\tilde{\mathrm{S}}^{(t)}\right).$$

$$\mathcal{M}_t = \mathcal{M}_{t-1}\left(y_t\right),$$

$$o_t = y_t \otimes \mathcal{M}^*_t\left(y_t\right)$$



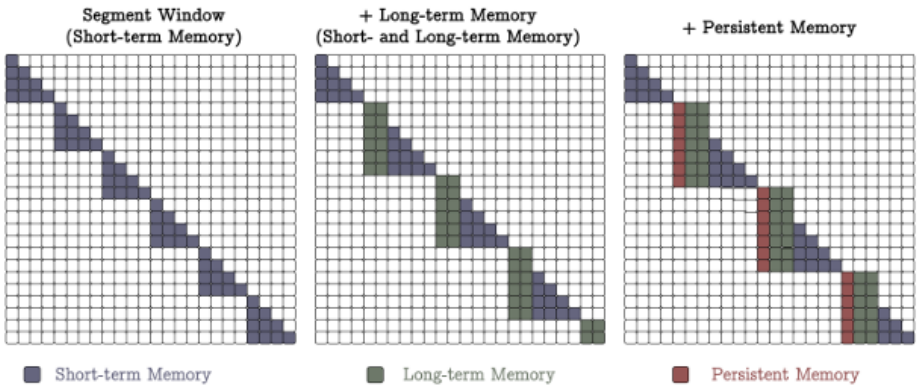
Memory as a Gate

$$\tilde{x} = \begin{bmatrix} p_1 & p_2 & \dots & p_{N_p} \end{bmatrix} \,||\, x,$$

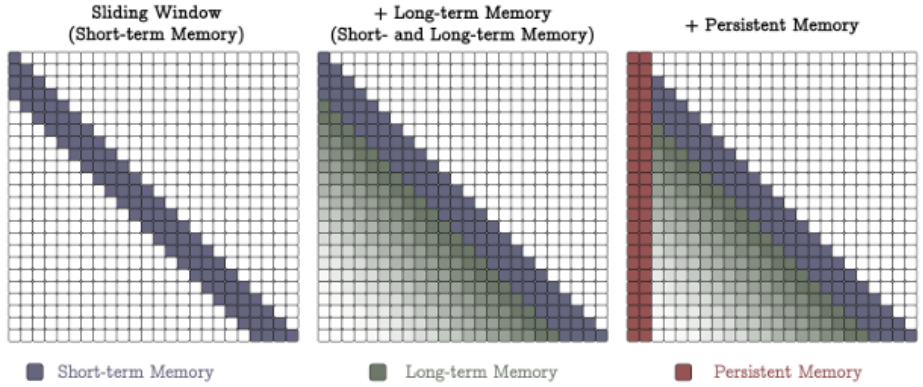
$$y = \mathtt{SW\text{-}Attn}^*\left(\tilde{x}\right),$$

$$o = y \otimes \mathcal{M}(\tilde{x}),$$

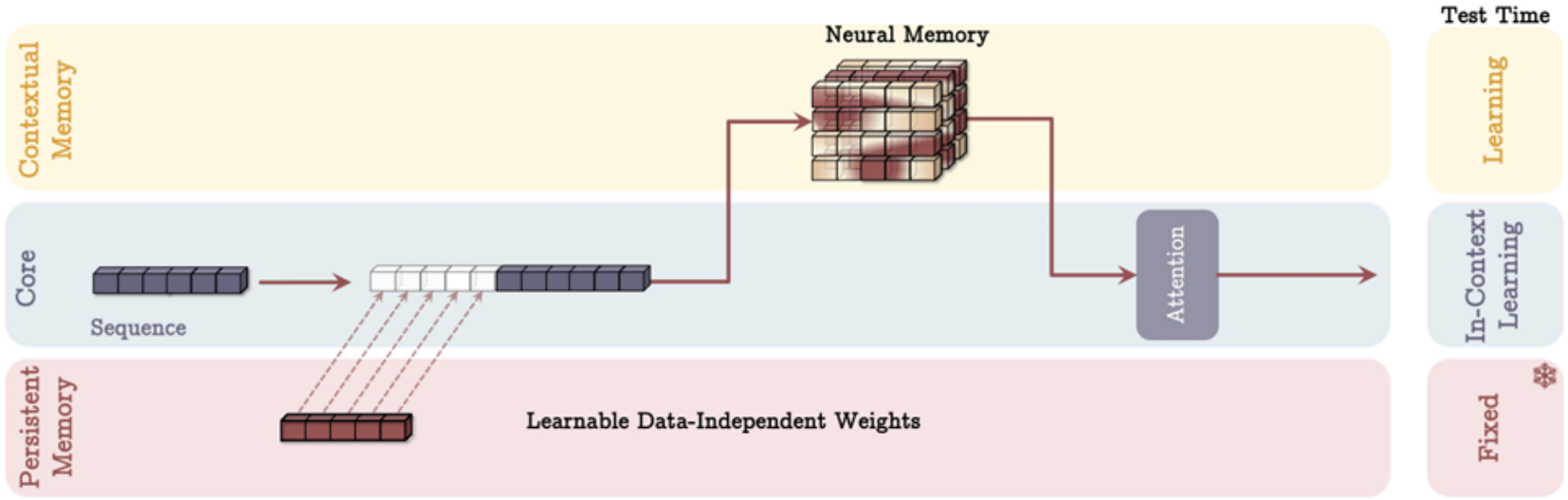
Titans: Learning to Memorize at Test Time



(a) **Memory as a Context (MAC).** We segment the sequence and use full causal attention in each window. Again, the first N_p tokens are persistent memory and the next N_l are long-term memory tokens



(b) **Memory as Gating (MAG).** We use sliding window attention (SWA) as a short-term memory and our neural memory module as a long-term memory, combining by a gating.



Memory as a Layer

$$\tilde{x} = \begin{bmatrix} p_1 & p_2 & \dots & p_{N_p} \end{bmatrix} \| x,$$
$$y = \mathcal{M}(\tilde{x}),$$
$$o = \texttt{SW-Attn}(y),$$

Titans: Learning to Memorize at Test Time

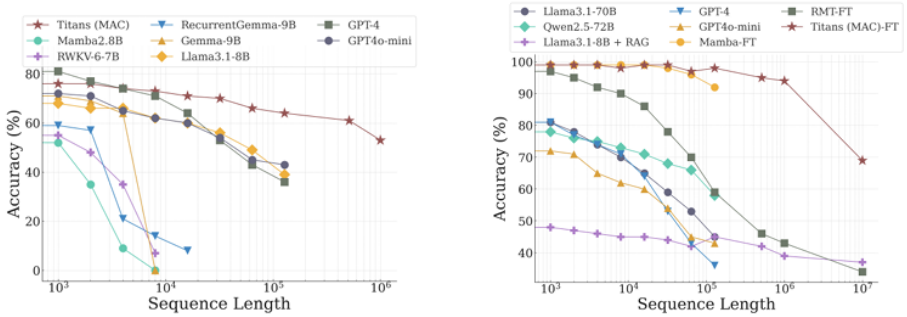
北京大学 PEKING UNIVERSITY

Table 1: Performance of Titans and recurrent- and Transformer-based baselines on language modeling and common-s reasoning tasks. Hybrid models are marked with *. The best results among simple and hybrid models are highligl

Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg. ↑
340M params / 15B tokens											
Transformer++	31.52	41.08	30.76	62.98	34.76	50.53	45.21	24.05	36.81	58.24	42.92
RetNet	32.50	49.73	28.24	62.61	34.15	50.91	44.27	23.62	36.79	59.72	42.54
GLA	28.51	43.02	28.73	64.05	35.96	50.00	54.19	24.29	37.13	58.39	44.09
Mamba	30.83	40.21	29.94	63.79	35.88	49.82	49.24	24.56	35.41	60.07	43.59
DeltaNet	28.65	47.30	28.43	63.52	35.95	49.63	52.68	25.37	37.96	58.79	44.04
TTT	27.44	34.19	30.06	63.97	35.71	50.08	53.01	26.11	37.32	59.83	44.51
Gated DeltaNet	27.01	30.94	34.11	63.08	38.12	51.60	55.28	26.77	34.89	59.54	45.42
Titans (LMM)	26.18	29.97	34.98	64.73	39.61	51.85	55.60	28.14	34.52	59.99	46.17
Titans (MAC)*	25.43	28.13	36.00	65.32	40.35	51.21	58.17	29.00	38.63	60.18	47.36
Titans (MAG)*	25.07	28.72	36.71	64.88	40.56	52.49	57.72	28.16	39.75	60.01	47.54
Titans (MAL)*	24.69	28.80	35.74	64.97	39.44	51.97	56.58	28.21	38.14	57.32	46.55
400M params / 15B tokens											
Transformer++	30.63	37.37	29.64	64.27	37.72	51.53	54.95	27.36	38.07	61.59	45.64
RetNet	29.92	46.83	29.16	65.23	36.97	51.85	56.01	27.55	37.30	59.66	45.47
HGRN2	32.33	47.14	26.12	64.52	35.45	52.24	55.97	25.51	37.35	59.02	44.52
GLA	27.96	36.66	27.86	65.94	37.41	49.56	56.01	26.36	38.94	59.84	45.24
Mamba	29.22	39.88	29.82	65.72	37.93	50.11	58.37	26.70	37.76	61.13	45.94
Mamba2	26.34	33.19	32.03	65.77	39.73	52.48	59.00	27.64	37.92	60.72	46.91
DeltaNet	27.69	44.04	29.96	64.52	37.03	50.82	56.77	27.13	38.22	60.09	45.57
TTT	26.11	31.52	33.25	65.70	39.11	51.68	58.04	28.99	38.26	59.87	46.86
Gated DeltaNet	25.47	29.24	34.40	65.94	40.46	51.46	59.80	28.58	37.43	60.03	47.26
Samba*	25.32	29.47	36.86	66.09	39.24	51.45	60.12	27.20	38.68	58.22	47.23
Gated DeltaNet-H2*	24.19	28.09	36.77	66.43	40.79	52.17	59.55	29.09	39.04	58.56	47.69
Titans (LMM)	25.03	28.99	35.21	65.85	40.91	52.19	59.97	29.20	38.74	60.85	47.83
Titans (MAC)*	25.61	27.73	36.92	66.39	41.18	52.80	60.24	29.69	40.07	61.93	48.65
Titans (MAG)*	23.59	27.81	37.24	66.80	40.92	53.21	60.01	29.45	39.91	61.28	48.60
Titans (MAL)*	23.93	27.89	36.84	66.29	40.74	52.26	59.85	29.71	38.92	58.40	47.87
760M params / 30B tokens											
Transformer++	25.21	27.64	35.78	66.92	42.19	51.95	60.38	32.46	39.51	60.37	48.69
RetNet	26.08	24.45	34.51	67.19	41.63	52.09	63.17	32.78	38.36	57.92	48.46
Mamba	28.12	23.96	32.80	66.04	39.15	52.38	61.49	30.34	37.96	57.62	47.22
Mamba2	22.94	28.37	33.54	67.90	42.71	49.77	63.48	31.09	40.06	58.15	48.34
DeltaNet	24.37	24.60	37.06	66.93	41.98	50.65	64.87	31.39	39.88	59.02	48.97
TTT	24.17	23.51	34.74	67.25	43.92	50.99	64.53	33.81	40.16	59.58	47.32
Gated DeltaNet	21.18	22.09	35.54	68.01	44.95	50.73	66.87	33.09	39.21	59.14	49.69
Samba*	20.63	22.71	39.72	69.19	47.35	52.01	66.92	33.20	38.98	61.24	51.08
Gated DeltaNet-H2*	19.88	20.83	39.18	68.95	48.22	52.57	67.01	35.49	39.39	61.11	51.49
Titans (LMM)	20.04	21.96	37.40	69.28	48.46	52.27	66.31	35.84	40.13	62.76	51.56
Titans (MAC)	19.93	20.12	39.62	70.46	49.01	53.18	67.86	36.01	41.87	62.05	52.51
Titans (MAG)	18.61	19.86	40.98	70.25	48.94	52.89	68.23	36.19	40.38	62.11	52.50
Titans (MAL)	19.07	20.33	40.05	69.99	48.82	53.02	67.54	35.65	30.98	61.72	50.97

Table 2: Performance of Titans and baselines on S-NIAH task from RULER benchmark. The best results among simple and hybrid models are highlighted.

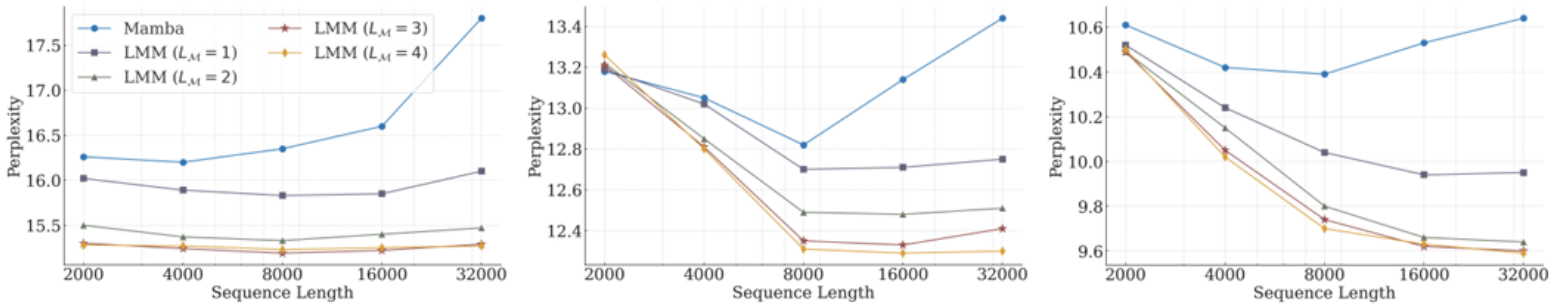
Model	S-NIAH-PK				S-NIAH-N				S-NIAH-W			
	2K	4K	8K	16K	2K	4K	8K	16K	2K	4K	8K	16K
TTT	98.4	98.8	98.0	88.4	60.2	36.6	10.2	4.4	78.8	28.0	4.4	0.0
Mamba2	98.6	61.4	31.0	5.4	98.4	55.8	14.2	0.0	42.2	4.2	0.0	0.0
DeltaNet	96.8	98.8	98.6	71.4	47.2	15.4	12.8	5.4	46.2	20.0	1.6	0.0
Titans (LMM)	99.8	98.4	98.2	96.2	100.0	99.8	93.4	80.2	90.4	89.4	85.8	80.6
Titans (MAC)	99.2	98.8	99.0	98.4	99.6	98.2	97.6	97.4	98.2	98.2	95.6	95.2
Titans (MAG)	99.4	98.0	97.4	97.4	99.2	98.8	97.2	98.6	98.0	98.0	90.2	88.2
Titans (MAL)	98.8	98.6	98.8	97.8	99.8	98.1	96.8	96.4	98.0	97.4	92.0	90.4



(a) Few-shot Setup (b) Fine-Tuning Setup

Figure 6: Performance of Titans and baselines on BABILong benchmark. Titans (MAC) outperforms all baselines, including extremely large models, e.g., GPT4.

Titans: Learning to Memorize at Test Time



(a) 170M Parameters (b) 360M Parameters (c) 760M Parameters

Figure 7: The effect of memory depth on the perplexity. Deeper long-term memory results in better scaling in longer sequences.

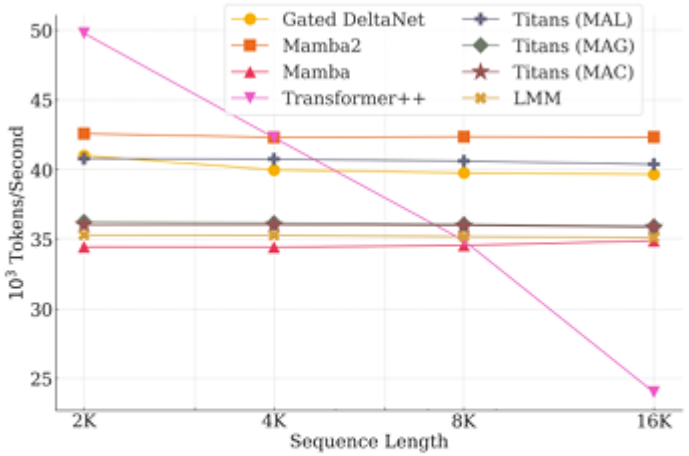


Figure 9: Training throughput comparison of Titans and baselines.

Table 5: Ablation Study on Titans. All components of Titans are positively contributing to its performance.

Model	Language Modeling ppl ↓	Reasoning acc ↑	Long Context acc ↑
LMM	27.01	47.83	92.68
+Attn (MAC)	26.67	48.65	97.95
+Attn (MAG)	25.70	48.60	96.70
+Attn (MAL)	25.91	47.87	96.91
Linear Memory	28.49	46.97	85.34
w/o Convolution	28.73	45.82	90.28
w/o Momentum	28.98	45.49	87.12
w/o Weight Decay	29.04	45.11	85.60
w/o Persistent Memory	27.63	46.35	92.49

MoM: Linear Sequence Modeling with Mixture-of-Memories

Shanghai AI Laboratory

MoM: Linear Sequence Modeling with Mixture-of-Memories

Method	Memory Update Rule
Linear Attn	$\boldsymbol{M}_t = \boldsymbol{M}_{t-1} + \boldsymbol{k}_t^T \boldsymbol{v}_t$
Lightning	$\boldsymbol{M}_t = \gamma \boldsymbol{M}_{t-1} + \boldsymbol{k}_t^T \boldsymbol{v}_t$
RetNet	$\boldsymbol{M}_t = \gamma \boldsymbol{M}_{t-1} + \boldsymbol{k}_t^T \boldsymbol{v}_t$
GLA	$\boldsymbol{M}_t = (\boldsymbol{a}_t^T \boldsymbol{1}) \boldsymbol{M}_{t-1} + \boldsymbol{k}_t^T \boldsymbol{v}_t$
DeltaNet	$\boldsymbol{M}_t = (\boldsymbol{I} - \boldsymbol{k}_t^T \boldsymbol{k}_t) \boldsymbol{M}_{t-1} + b_t \boldsymbol{k}_t^T \boldsymbol{v}_t$
G-DeltaNet	$\boldsymbol{M}_t = a_t (\boldsymbol{I} - \boldsymbol{k}_t^T \boldsymbol{k}_t) \boldsymbol{M}_{t-1} + b_t \boldsymbol{k}_t^T \boldsymbol{v}_t$
Mamba2	$\boldsymbol{M}_t = a_t \boldsymbol{M}_{t-1} + b_t \boldsymbol{k}_t^T \boldsymbol{v}_t$
HGRN2	$\boldsymbol{M}_t = (\boldsymbol{a}_t^T \boldsymbol{1}) \boldsymbol{M}_{t-1} + (1 - \boldsymbol{a}_t)^T \boldsymbol{v}_t$
TTT	$\boldsymbol{M}_t = \boldsymbol{M}_{t-1} + b_t \nabla l(\boldsymbol{M}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t)$
Titans	$\boldsymbol{M}_t = a_t \boldsymbol{M}_{t-1} + b_t \nabla l(\boldsymbol{M}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t)$

Table 1: **Memory Update Rules.** We demonstrate that several current linear sequence models can be viewed as recurrent models in terms of memory updates, where $a_t, b_t \in (0, 1)$ are data-dependent scaler, $\boldsymbol{a}_t$ is data-dependent vector, and γ is a data-independent constant.

$$\boldsymbol{M}_t^m = \gamma \boldsymbol{M}_{t-1}^m + (\boldsymbol{k}_t^m)^T \boldsymbol{v}_t^m \in \mathbb{R}^{d \times d},$$

$$\tilde{\boldsymbol{M}}_t = \sum_m g_t^{(m)} \boldsymbol{M}_t^m \in \mathbb{R}^{d \times d},$$

$$\boldsymbol{o}_t = \boldsymbol{q}_t \tilde{\boldsymbol{M}}_t \in \mathbb{R}^d,$$

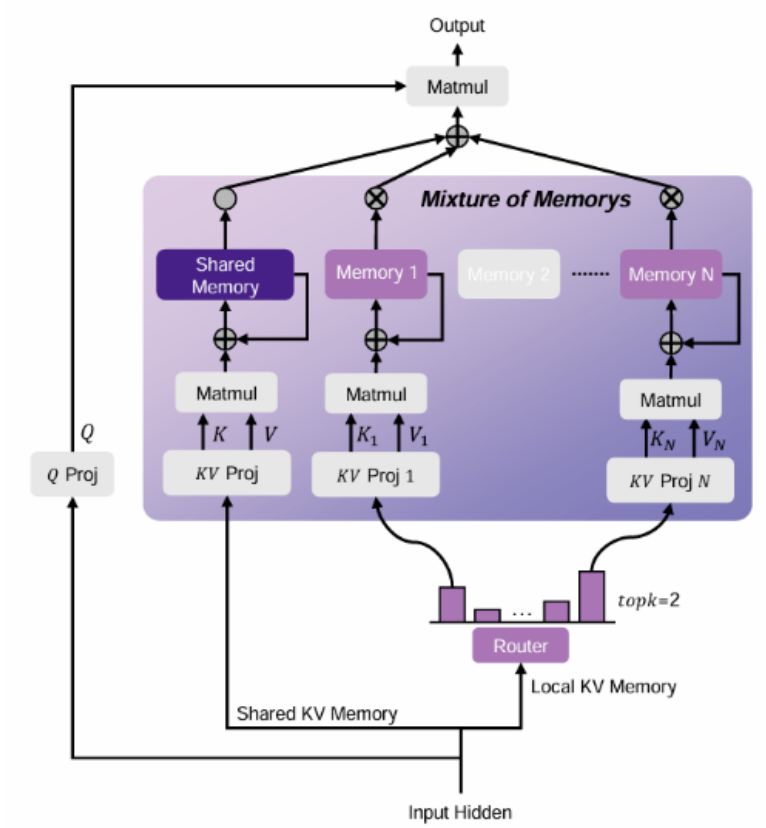


Figure 1: **Framework of MoM.** Each input token selectively activates and updates K memory states, leaving non-activated memory states unchanged to avoid interference from current input. Additionally, we introduce a continuously activated shared memory. This figure presents the basic memory update mechanism; other mechanisms involving gating or more complex updates follow a similar approach.

MoM: Linear Sequence Modeling with Mixture-of-Memories

北京大学 PEKING UNIVERSITY

Scale	Model	FDA	SWDE	SQUAD	NQ	TriviaQA	Drop	Avg.
340M Params	Transformer++	46.14	25.87	33.22	18.94	45.97	20.03	31.70
15B Tokens	RetNet	5.90	9.28	22.41	6.91	40.05	18.59	17.19
L=24, d=1024	HGRN2	11.53	17.34	24.08	12.67	43.84	17.35	21.14
	GLA	11.26	16.78	27.85	12.77	43.90	17.68	21.71
	GSA	6.36	16.87	21.90	14.60	42.18	16.72	19.77
	Gated DeltaNet	20.53	23.24	28.55	**14.98**	**44.91**	16.48	24.78
	MoM	**30.79**	**26.05**	**29.63**	13.84	44.79	**20.41**	**27.59**
1.3B Params	Transformer++†	44.32	32.43	42.59	24.49	58.47	21.56	37.31
100B Tokens	RetNet†	13.62	22.59	33.46	15.43	53.79	19.79	26.45
L=24, d=2048	HGRN2†	12.35	23.24	33.19	19.10	55.27	19.65	27.13
	GLA†	27.61	30.93	35.04	22.27	56.28	19.45	31.93
	GSA†	23.25	32.80	35.57	22.96	57.05	20.65	32.05
	Gated DeltaNet	30.25	27.65	34.06	23.22	58.23	20.36	32.30
	MoM	**41.14**	**34.30**	**37.08**	**24.11**	**58.59**	**21.03**	**36.04**

le 2: **Results on Recall-Intensive Tasks.** All inputs are truncated to a maximum length of 2K tokens. MoM ificantly outperforms all other linear models across both model sizes. In the 1.3B model, MoM even achieves formance very close to that of Transformer models.

Scale	Model	Wiki. ppl↓	Lamb. ppl↓	ARC-e acc↑	ARC-c acc_n↑	Hella. acc_n↑	Lamb. acc↑	PIQA acc↑	Wino. acc↑	Avg.
340M Params	Transformer++	26.88	76.46	44.91	**25.94**	34.95	26.90	64.31	51.07	41.35
15B Tokens	RetNet	31.07	87.11	44.49	23.04	33.86	23.93	63.49	52.33	40.19
L=24, d=1024	HGRN2	27.90	77.40	45.24	23.63	35.61	24.74	65.45	**54.06**	41.46
	GLA	28.78	79.95	44.53	22.27	34.84	24.94	63.93	51.38	40.32
	GSA	28.17	82.50	45.50	24.23	35.00	24.02	64.85	50.43	40.67
	Gated DeltaNet	26.47	58.59	46.04	23.55	35.18	27.01	**66.05**	50.83	41.44
	MoM	**26.00**	**51.25**	**46.13**	24.15	**35.91**	**28.26**	65.61	52.57	**42.11**
1.3B Params	Transformer++†	17.61	19.29	55.01	28.07	49.21	40.95	70.08	56.27	49.93
100B Tokens	RetNet†	18.18	21.97	57.49	26.88	48.09	37.75	69.37	53.28	48.81
L=24, d=2048	HGRN2†	17.32	15.65	**58.33**	28.07	**51.93**	42.31	71.33	52.01	50.66
	GLA†	17.61	19.66	55.18	27.56	48.89	40.03	69.86	53.91	49.24
	GSA†	16.69	16.02	**58.33**	**28.33**	50.98	42.03	**72.25**	53.43	50.89
	Gated DeltaNet	17.14	18.80	56.82	27.39	49.77	39.94	71.76	51.78	49.58
	MoM	**16.64**	**14.83**	55.35	27.99	50.95	**43.43**	71.27	**56.83**	**50.97**

Table 3: **Results on Common-Sense Reasoning Tasks.** The performance of linear models and Transformer models is comparable; however, MoM consistently achieves the best average performance across all model sizes.

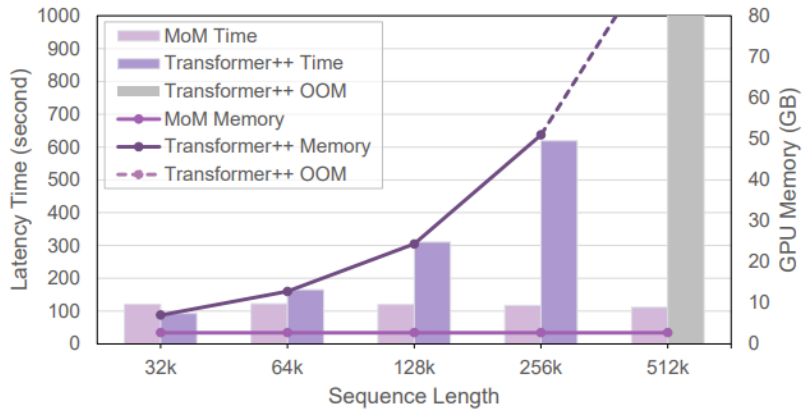


Figure 2: **Efficiency of MoM.** We demonstrate the inference time and GPU memory consumption required to generate 1K tokens at specific sequence lengths.