Titans: Learning to Memorize at Test Time

Ali Behrouz[†], Peilin Zhong[†], and Vahab Mirrokni[†]

[†]Google Research

HUMANE Lab 박현빈

25.02.07

Background

- RNN compresses data into a fixed size, but its performance is not good.
- Attention achieves good performance due to contextualization between tokens, but its context window is limited, preventing long-form memory retention and incurring quadratic cost.
- Therefore, many hybrid models are emerging that use Attention as short-form memory and memory module as long-form memory.
- By storing older context in the long-term memory module and allowing Attention to focus only on recent context, computational costs can be reduced.

Background

- RNN can be defined as models with a vector-valued memory module (hidden state)
- Key and value matrices of Transformers acts as the model's memory

Contributions and Roadmap

- Neural Memory
  - deep neural long-term memory learns how to memorize/store the data into its parameters at test time
  - Unexpected inputs are remembered more effectively.
  - The surprise level of an input is measured using the gradient of the neural network.
  - A decaying mechanism is applied to efficiently manage limited memory by forgetting less relevant information.
  - To enable faster memory updates at test time, a parallelizable algorithm is proposed.

Contributions and Roadmap

- Titans architectures
  - core : short-term memory
  - long-term memory : neural long-term memory module
  - persistent memory : learnable but data-independent parameters

- Incorporate memory as (i) a context, (ii) a layer, (iii) a gated branch

Preliminaries

- Attention

$$\mathbf{Q} = x\mathbf{W}_{\mathrm{Q}}, \qquad \mathbf{K} = x\mathbf{W}_{\mathrm{K}}, \qquad \mathbf{V} = x\mathbf{W}_{\mathrm{V}},$$

$$\mathbf{y}_i = \sum_{j=1}^{i} \frac{\exp\left(\mathbf{Q}_i^{\top}\mathbf{K}_j/\sqrt{d_{\mathrm{in}}}\right)\mathbf{V}_j}{\sum_{\ell=1}^{i}\exp\left(\mathbf{Q}_i^{\top}\mathbf{K}_\ell/\sqrt{d_{\mathrm{in}}}\right)},$$

- Efficient Attentions (Linear Attentions)
  - softmax 대신 다음과 같은 kernel function 사용$\phi(x,y) = \phi(x)\phi(y)$

$$\mathbf{y}_i = \sum_{j=1}^{i} \frac{\phi(Q_i^{\top}K_j)}{\sum_{\ell=1}^{i}\phi(Q_i^{\top}K_\ell)}V_j = \sum_{j=1}^{i} \frac{\phi(Q_i)^{\top}\phi(K_j)}{\sum_{\ell=1}^{i}\phi(Q_i)^{\top}\phi(K_\ell)}V_j = \frac{\phi(Q_i)^{\top}\sum_{j=1}^{i}\phi(K_j)V_j}{\phi(Q_i)^{\top}\sum_{\ell=1}^{i}\phi(K_\ell)},$$

$$\mathcal{M}_t = \mathcal{M}_{t-1} + K_t^{\top}V_t\,,$$
$$\mathbf{y}_t = Q_t\mathcal{M}_t\,,$$

Preliminaries

- Modern linear models
  - hidden state of RNN can be treated as a memory unit
  - recurrence process can be split into the read and write operations in the memory unit
  - f(., .) is the read and g(., .) is the write

    $\mathcal{M}_t = f(\mathcal{M}_{t-1}, x_t),$

    $\mathbf{y}_t = g(\mathcal{M}_t, x_t),$
  - Linear Transformer is equivalent to additively compress and write keys and values into a matrix-valued memory unit M_t
  - but when dealing with long context data, this additive process results in memory overflow
  - to address this, several studies have presented forgetting gate mechanisms, where it can erase the memory when it is needed

Preliminaries

- Memory modules
  - memory has always been one of the core parts of the neural network designs
  - memory modules, however, are based on momentary surprise, missing the token flow in the sequences

Learning to Memorize at Test Time

- memorization has been known as an undesirable phenomena in neural networks as it limits the model generalization
- memorization of the training data might not be helpful at test time, in which the data might be out-of-distribution
- to ensure that the long-term memory module stores only valuable information, the model needs to learn how to memorize/forget data at test time

Learning to Memorize at Test Time

- The neural memory module M_t aims to compress the past information into its parameters
- Definition of surprise for a model can be its gradient

$$\mathcal{M}_t = \mathcal{M}_{t-1} - \theta_t \underbrace{\nabla \ell(\mathcal{M}_{t-1}; x_t)}_{\text{Surprise}}.$$

- Surprise is broken into (1) past surprise and (2) momentary surprise

$$\mathcal{M}_t = \mathcal{M}_{t-1} + S_t,$$

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

- A loss function for the model to learn the associations between key and value $\ell(\mathcal{M}_{t-1}; x_t) = \|\mathcal{M}_{t-1}\left(\mathbf{k}_t\right) - \mathbf{v}_t\|_2^2$

Learning to Memorize at Test Time

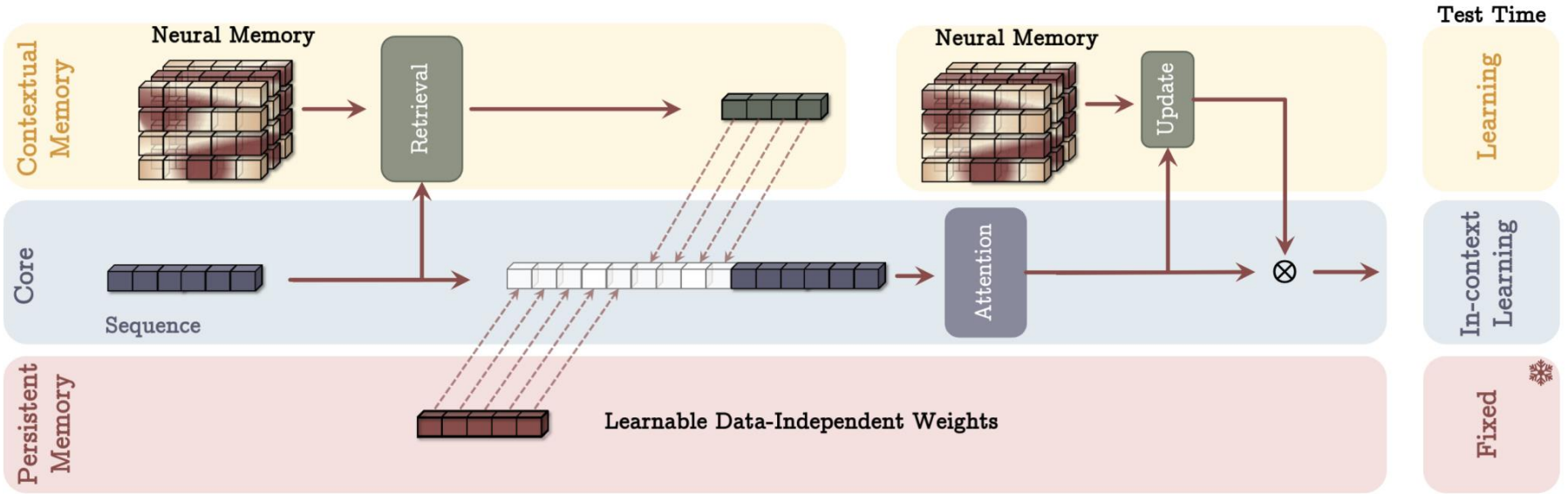
- Forgetting Mechanism

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

- Simple MLPs with $L_M \geq 1$ layers as the architecture of long-term memory
- Model simply uses the forward pass without weight update to retrieve a memory corresponding to a query

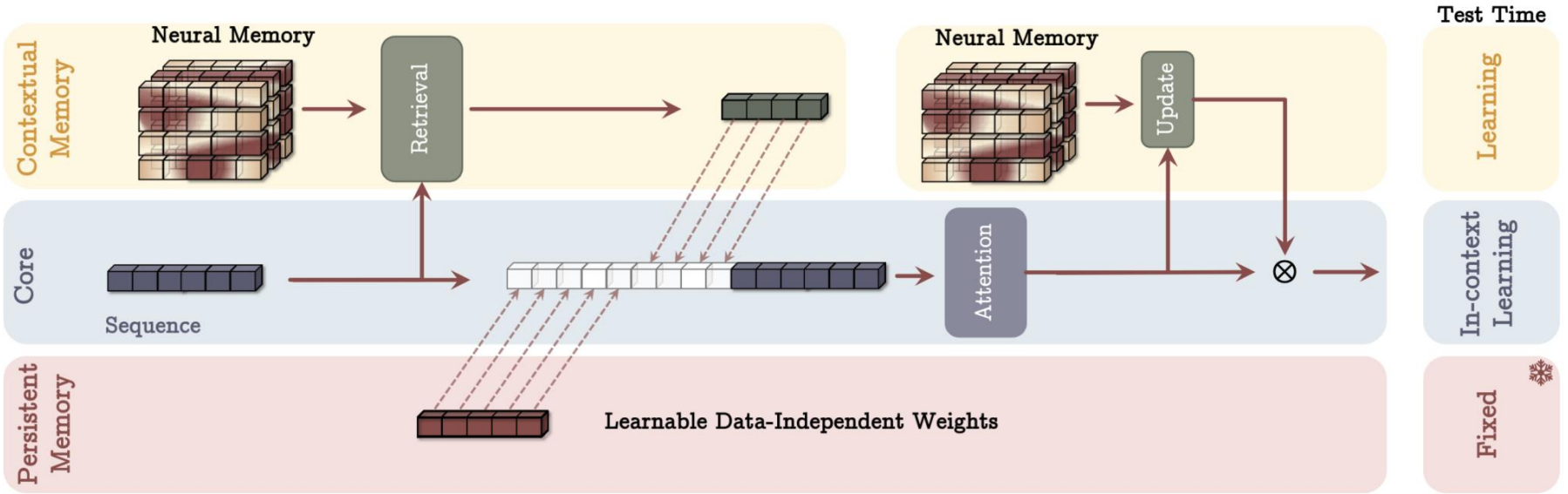
$$y_t = \mathcal{M}^*(\mathbf{q}_t)$$

Learning to Memorize at Test Time



- Persistent memory is task-related memory, not input-dependent memory
- Persistent memory store the abstraction of the task knowledge (how the task can be done)

MAC(Memory as a Context)



Retrieval

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

Attention

$$\tilde{S}^{(t)} = [p_1 \quad p_2 \quad \dots \quad p_{N_p}] \;||\; h_t \;||\; S^{(t)}$$

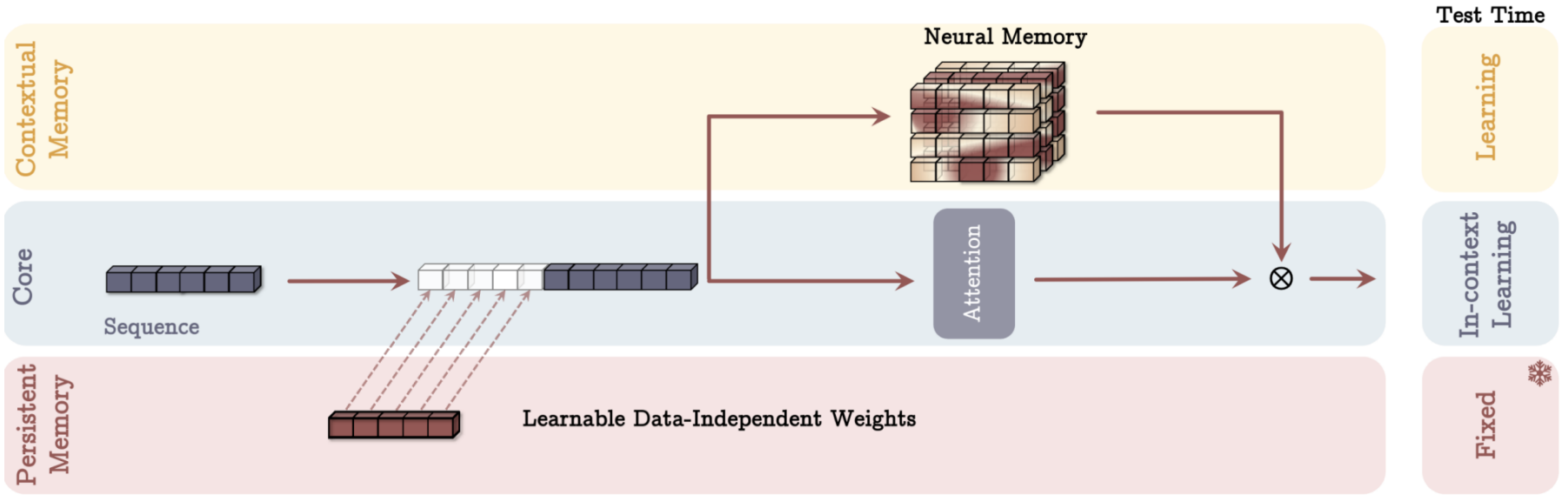
$$y_t = \texttt{Attn}\left(\tilde{S}^{(t)}\right).$$

Final output

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

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

MAG(Memory as a Gate)



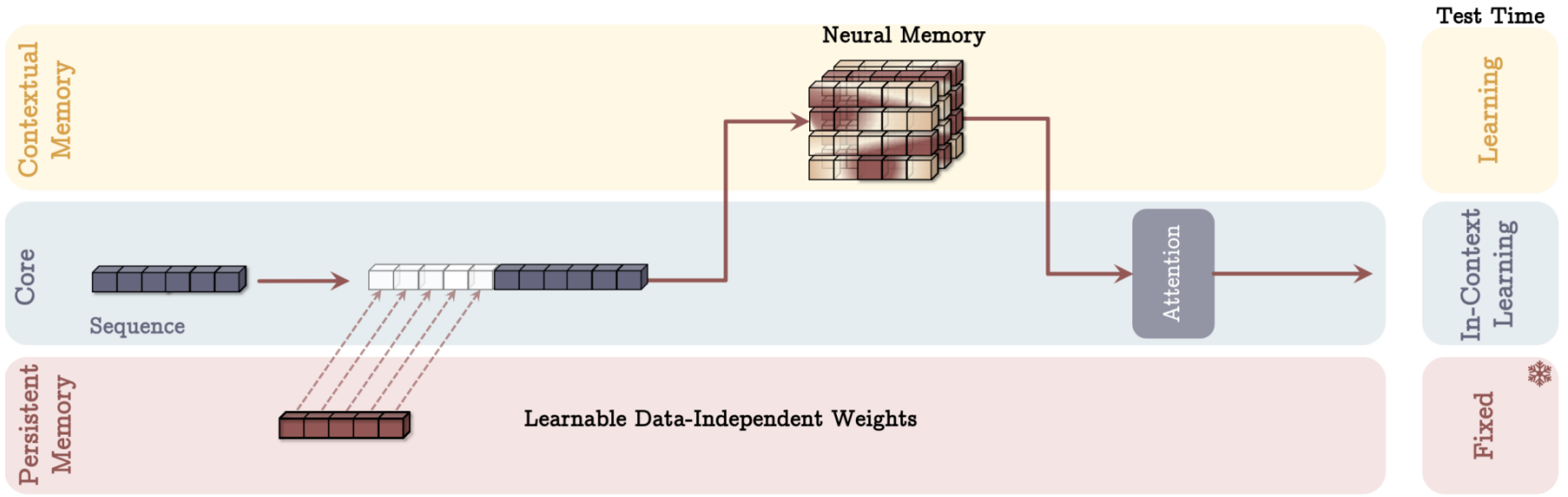
Attention

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

Final output

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

MAL(Memory as a Layer)



Final output

$$o = \text{SW-Attn}\,(y)$$

Architecture Details

- Using residual connections in all blocks
- Using SiLU activation as the non-linear activation for computing query, key, values
- Using L2-norm to normalize queries and keys
- Using 1D depthwise-separable convolution layer after each of the query, key, and value projections

Experiments

- Models : 170M, 340M, 400M, 760M
- While the first three models trained on 15B tokens sampled from FineWeb-Edu dataset, the last one is trained on 30B tokens
- Tokenizer : LLama2 tokenizer with a vocabulary size of 32K

Experiments

- Language Modeling

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

  - MAG and MAC outperform MAL
  - comparing MAG and MAC, MAC performs better when dealing with longer dependencies in the data

Experiments

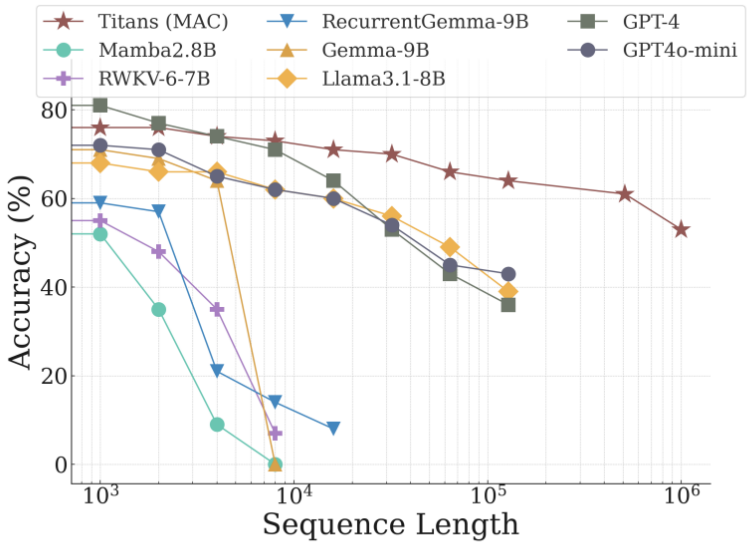
- Needle in a Haystack (NIAH)

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

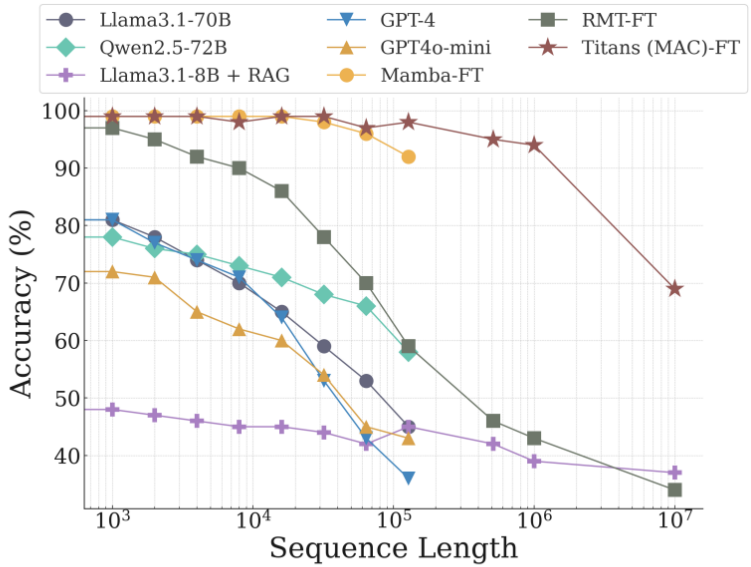
  - evaluate the model on retrieving a piece of information from long texts (2k ~ 16k)

Experiments

- BABILong



(a) Few-shot Setup

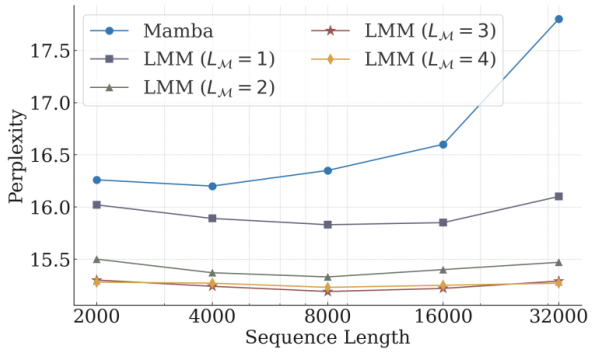


(b) Fine-Tuning Setup

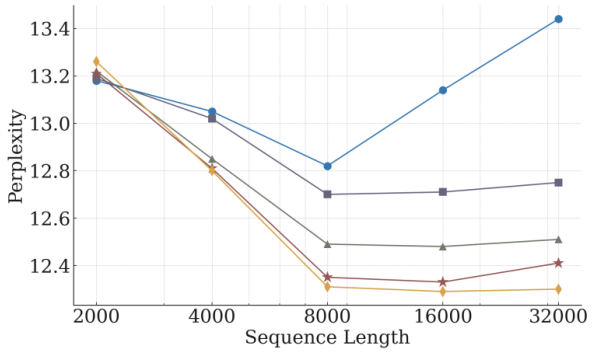
- the model needs to reason across facts distributed in extremely long documents

Experiments

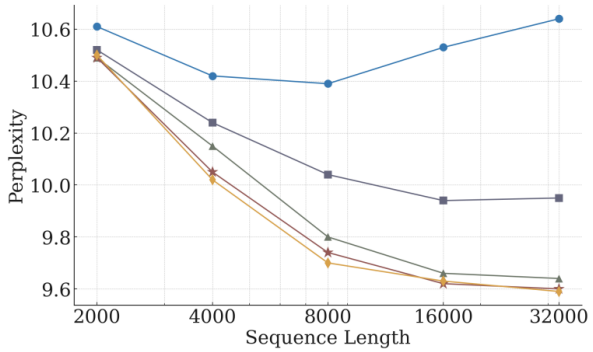
- The effect of deep memory



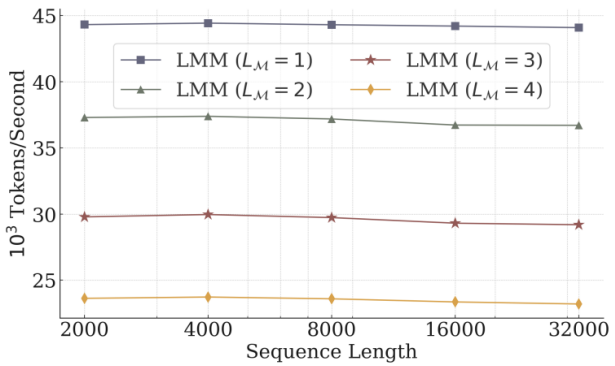
(a) 170M Parameters



(b) 360M Parameters



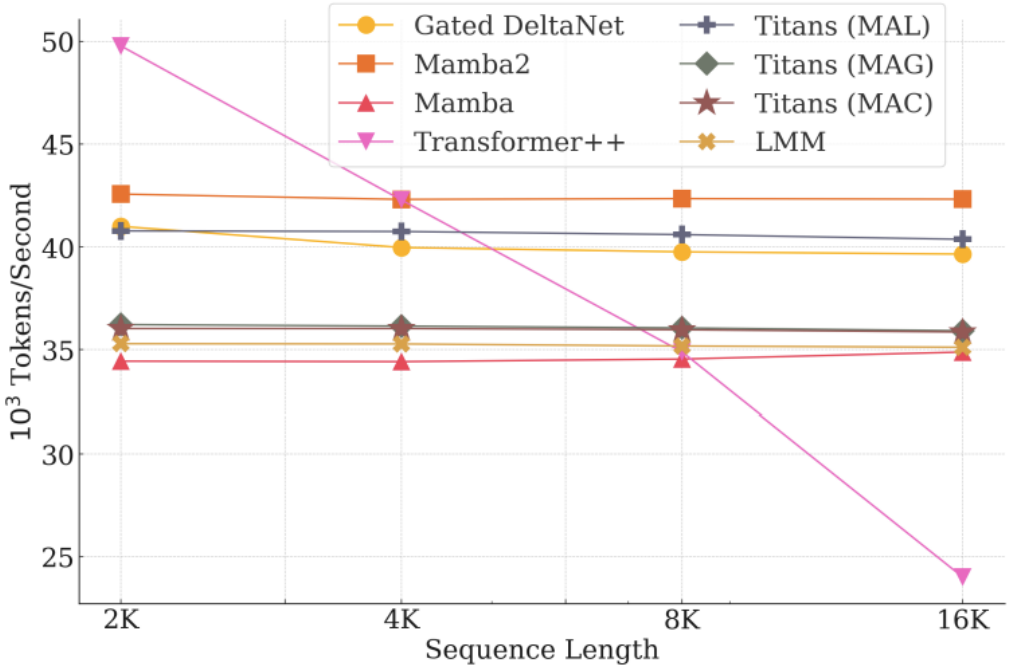
(c) 760M Parameters



- As the number of layers increase, the performance improve, whereas the throughput decrease

Experiments

- Efficiency



Experiments

- Ablation study
  - MAC >= MAG > MAL
  - weight decay > momentum > convolution > persistent memory

My Review

- Models such as memory modules and Linear Transformers have been continuously researched to overcome the limitations of Transformers
- Titan performs well in long-context and is also efficient
- However, it is expected to be weaker in areas such as math

Open Question

- How can the Titan model be made good at math?