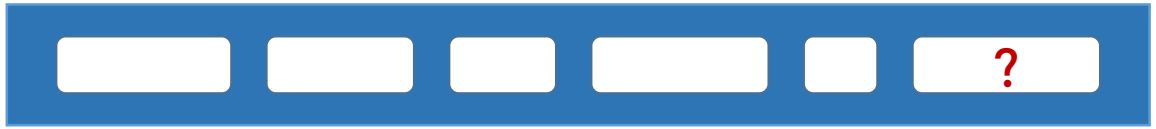
Titans: Learning to Memorize at Test Time

25. 08.12

노정빈

I Motivation

❖ 기존 LLM(Transformer)의 한계점



- 앞에 나오는 토큰들을 전부 읽어 다음에 나올 단어를 추정하는 방식
- Context 길이가 길어질수록 시간 및 메모리 복잡도 증가
- Context window를 초과하는 범위를 입력하면 앞에 나온 것을 까먹음 -> 장기 기억X * Context window: 한 번에 병렬적으로 처리할 수 있는 토큰 수

➢ 이러한 한계를 극복하기 위해 새로운 신경망 장기 메모리 모듈(Neural Long-term Memory Module, LMM) 제안

❖ Titans

- Attention: 단기 메모리 역할
- LMM: 장기 메모리 역할
- Persistent Memory: 영구 메모리 역할

➢ 세 모듈을 기반으로 “Titans”라는 새로운 Architecture를 소개하고, 메모리를 효과적으로 통합하는 세 가지 변형을 제시

Ⅱ Backgrounds

❖ Classical Attention

- 입력 $x \in \mathbb{R}^{N \times d_{in}}$ 이 주어지면 Q, K, V 행렬에 대한 softmax 함수를 기반으로 출력 y $\in \mathbb{R}^{N \times d_{in}}$ 계산

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

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

y_i: i번째 토큰의 출력. 이전 토큰들과 i번째 토큰의 정보를 종합하여 생성

exp(): 지수함수

$Q_i^T K_j$: Q_i와 K_j 내적

d_{in}: 입력 임베딩 차원. 불안정 학습 방지

Q_i, K_i, V_i: i번째 토큰의 query. key, value 벡터

- 출력을 계산하기 위해 최소 $N \times d$연산자가 필요하므로 컨텍스트가 길수록 더 큰 메모리 소비, 낮은 처리량

➢ 이를 개선하기 위해 여러 변형의 Attention 개발

➢ 본 논문에선 kernel-based linear attention에 초점

Ⅱ Backgrounds

❖ Kernel-based Linear Attention

- 기존 attention에서 softmax 함수 대신 커널함수 $\phi()$로 대체하여 효율성을 높인 모듈
- $\phi()$는 다음과 같은 조건을 만족하는 어떤 함수(ReLU, 지수함수 등)
- $\phi(xy) = \phi(x)\phi(y)$

<Classical Attention>

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

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

- Classic은 매 시점 i마다 모든 $Q_i^T K_j$를 계산해야 했지만,

  Linear는 $\phi(K_j)V_j$와 $\phi(K_\ell)$의 누적 합으로 계산할 수 있기 때문에 훨씬 효율적

Ⅱ Backgrounds

❖ Kernel-based Linear Attention

- 누적 합 형식을 다음과 같은 순환 형식으로 변환할 수 있음

$$\mathbf{y}_i = \frac{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j) V_j}{\phi(Q_i)^\top \sum_{\ell=1}^{i} \phi(K_\ell)}, \longrightarrow \begin{aligned} \mathcal{M}_t &= \mathcal{M}_{t-1} + K_t^\top V_t, \\ \mathbf{y}_t &= Q_t \mathcal{M}_t, \end{aligned}$$

- 본 논문에서는 커널함수를 항등행렬 $\Phi(x) = Ix = x$로 정의
- 분모는 정규화 역할이므로 생략 가능
- $\mathcal{M}_t$: 누적된 key-value 쌍을 저장하는 메모리 유닛

➢ 이를 차용하여 신경망 장기 메모리의 업데이트 방식을 정의

Ⅲ 신경망 장기 메모리(Neural Long-term Memory Module, LMM)

❖ 핵심 아이디어

- 인간은 예상을 위반하는 사건을 더 잘 기억한다는 것에서 영감을 받아,
  '놀라움(surprise)'의 정도가 높은 입력을 더 잘 기억하도록 설계
- 테스트 타임에서도 데이터를 학습하고 기억하도록 설계

❖ 학습 프로세스 및 surprise 측정

- Surprise를 입력에 대한 기울기로 정의. 기울기가 클수록 입력 데이터가 과거 데이터와 다름
- 큰 surprise 후에 중요한 정보가 누락되는 문제가 있을 수 있어 surprise를 2가지로 나눔

(1) Past surprise: 가장 최신 과거의 정보가 얼마나 surprise 했는지의 정도

(2) Momentary surprise: 입력 데이터의 surprise 정도

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

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

η_t: 데이터에 따라 past surprise를 얼마나 반영할지를 제어

θ_t: 데이터에 따라 momentary surprise를 얼마나 반영할지를 제어

η_t->0: 과거의 surprise 무시(ex. 컨텍스트 변경)

η_t->1: 과거의 surprise 완전 통합(ex. 현재 토큰이 과거 토큰과 매우 높은 관련)

Ⅲ 신경망 장기 메모리(Neural Long-term Memory Module, LMM)

❖ **목적 함수**

- LMM이 테스트 타임에서도 학습하기 위해 연관 메모리로 작동하도록 학습
- 즉, 입력 x_t에 대한 키(k_t)와 값(v_t) 사이의 연관성을 학습하는 모델로 작동
- 이를 위한 손실 함수 $\ell(\cdot,\cdot)$ 정의

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

- LMM은 이 손실 함수를 최소화하는 방향으로 $\mathcal{M}_t$를 업데이트
- 현재 입력의 키에 대해 LMM이 생성하는 예측 값 $\mathcal{M}_{t-1}(k_t)$이 실제 값 v_t와 최대한 같아지도록 학습

Ex)

(1) 강아지(K) – 멍멍(V) 이라는 정보를 학습한 상태

(2) 메모리($\mathcal{M}_{t-1}$)에서 '강아지'라는 키 카드를 찾았을 때 메모리가 '야옹'이라고 예측($\mathcal{M}_{t-1}(k_t)$) 한다면, 실제 값 '멍멍'(v_t)과 다르므로 놀라게 됨(surprise). 이 놀람의 정도($\nabla\ell$)에 따라 메모리($\mathcal{M}$)를 업데이트

(3) Surprise를 통한 업데이트는 새로운 정보가 들어올 때마다 계속됨
즉, 메모리는 고정된 채로 정보를 찾는 것이 아니라 정보를 찾을 때마다(테스트 타임) 스스로 분류 체계와 저장 방식을 계속해서 개선

Ⅲ 신경망 장기 메모리(Neural Long-term Memory Module, LMM)

❖ Forgetting Mechanism

- 인간은 오래된 기억을 상세하게 기억하는 것이 아니라 중요한 정보만 추상화하여 기억
- 긴 시퀀스를 다룰 때 불필요한 과거 정보를 망각하는 것도 중요
- 이를 위해 적응형 망각 메커니즘(adaptive forgetting mechanism)을 사용하여 업데이트 규칙 수정

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

- α_t: 과거 데이터를 얼마나 잊어야 하는지 결정하는 gating mechanism
  α_t->0: 과거 데이터에 영향을 주지 않고 메모리 업데이트
  α_t->1: 전체 메모리 지움

- Gating Mechanism: Gate를 통해 입력된 정보와 이전 상태를 처리하여 현재 상태를 생성하는 메커니즘
  ex) 새로운 정보를 얼마나 수용할지, 과거 정보를 얼마나 잊을지

IV 영구 메모리(Persistent Memory)

❖ Persistent Memory

- LMM은 컨텍스트(입력)에 의존하는 메모리
- Persistent Memory는 입력에 의존하지 않는 메모리
- 즉, 학습은 가능하지만 입력과 상관없는 파라미터의 집합. 작업과 관련한 기본 지식들을 저장

Ex)

- LMM: 일기장 또는 개인 학습 노트

  -> 오늘 있었던 일, 만났던 사람 등 새롭게 접하는 구체적인 정보나 경험을 기억하고, 필요에 따라 업데이트
- Persistent Memory: 백과사전 또는 작업 매뉴얼

  -> “하늘은 푸르다”, “문장을 시작할 때는 대문자로” 등 특정 문맥에 따라 크게 변하지 않거나 일반적인 사실들

V Titans Architecture

❖ **기본 구성 요소**

- Titans는 다음과 같은 세 가지의 핵심 요소로 구성

(1) Core (단기 기억)

: Attention 메커니즘을 사용하여 현재 입력 시퀀스의 주요 정보 흐름을 처리하는 단기 기억 역할

제한된 context window 내에서 작동

(2) Contextual Memory (장기 기억)

: LMM을 사용하여 과거의 정보들을 기억/저장하는 장기 기억 역할

테스트 시점에서도 스스로 학습하여 지속적으로 정보를 추상화하고 파라미터에 인코딩

(3) Persistent Memory (영구 기억)

: 학습 가능하지만 입력 시퀀스와 관련 없는 데이터를 저장하는 영구 기억 역할

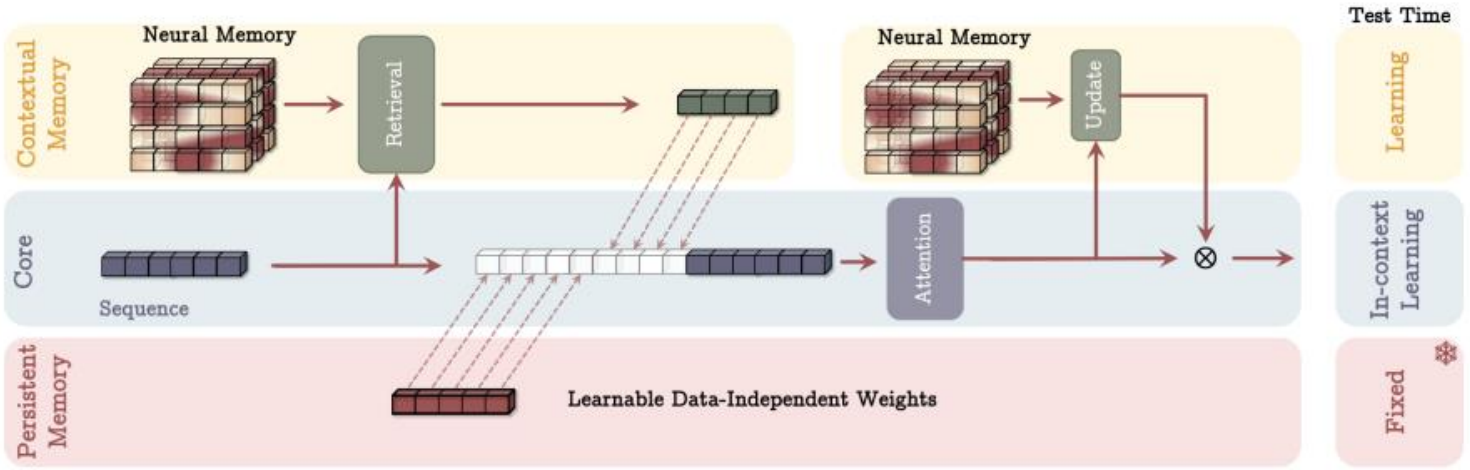
특정 작업에 대한 일반적인 지식, 변하지 않는 보편적인 지식을 인코딩

➢ 이 요소들의 상호작용 방식에 따라 총 세 가지의 Titans Architecture 제시

V Titans Architecture

*: 가중치 조정 없이 forward pass만 수행
‖·‖: concatenate
⊗: 비선형 gate: sigmoid 같은 비선형 활성화 함수를 사용한 gate

❖ MAC (Memory As a Context)



$$h_t = \mathcal{M}^*_{t-1}(\mathbf{q}_t),$$
$$\tilde{S}^{(t)} = \left[p_1 \quad p_2 \quad \cdots \quad p_{N_p}\right] \,\|\, h_t \,\|\, S^{(t)},$$
$$y_t = \text{Attn}\left(\tilde{S}^{(t)}\right).$$
$$\mathcal{M}_t = \mathcal{M}_{t-1}\left(y_t\right),$$
$$o_t = y_t \otimes \mathcal{M}^*_t\left(y_t\right).$$

- 장기 기억 모듈과 영구 기억 모듈을 단기 기억 모듈, 즉 Attention 메커니즘의 '추가적인 컨텍스트'로 활용하는 방식

  (1) 입력 시퀀스 x를 고정된 크기의 $S^{(i)}(i = 1, \ldots, \frac{N}{C})$로 분할

  (2) LMM에 현재 세그먼트($S^{(t)}$)가 들어오면 이를 쿼리로 변환하여 정보 검색

  (3) 현재 세그먼트에 검색된 정보와 영구 기억을 연결(concatenate)하여 새로운 세그먼트 구성

  (4) 새로 구성된 세그먼트를 attentio의 입력으로 사용하여 출력 생성

  (5) Attention의 출력을 바탕으로 LMM 업데이트

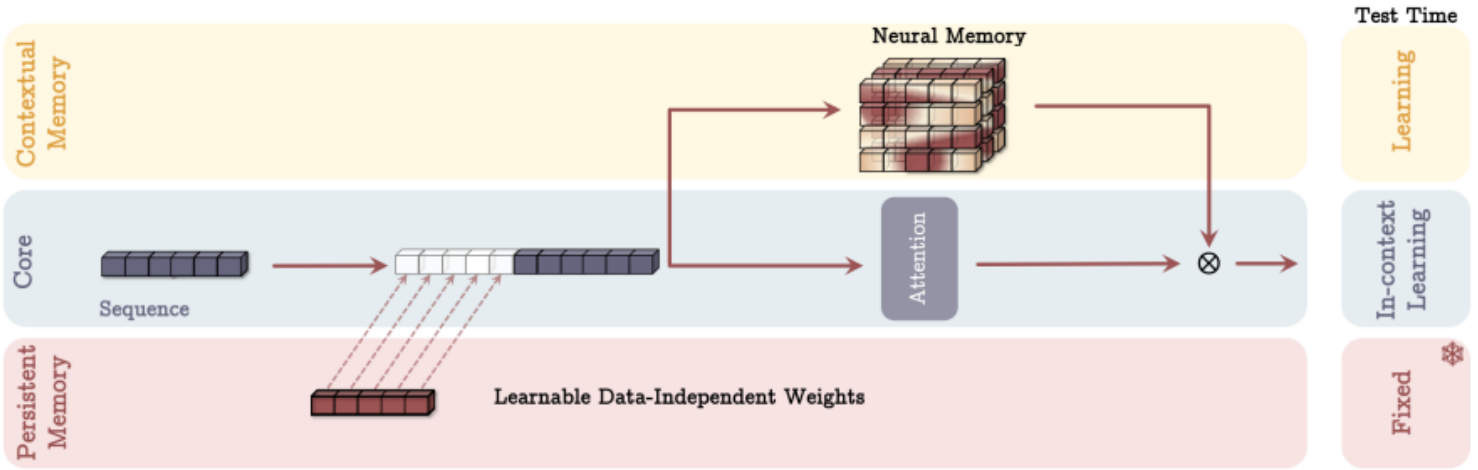
  (6) attention의 출력과 업데이트된 LMM의 정보를 결합하여 최종 출력 생성

- 장점: Attention이 LMM의 정보가 필요한지 능동적으로 판단하고, LMM이 유용한 정보만 저장하도록 도움

V Titans Architecture

SWA(Sliding Window Attention)
: context window 크기를 제한한 후, 그 내에서만 attention을 수행하게 하여 계산 효율성과 단기 기억의 정확성을 높이는 방법

❖ MAG (Memory As a Gate)



$$\tilde{x} = \begin{bmatrix} p_1 & p_2 & \cdots & p_{N_p} \end{bmatrix} \| x,$$
$$y = \text{SW-Attn}^*(\tilde{x}),$$
$$o = y \otimes \mathcal{M}(\tilde{x}),$$

- 장기 기억 모듈과 단기 기억 모듈이 병렬적으로 작동하며, 그 결과를 gating 메커니즘을 통해 결합하는 방식

  (1) 입력 시퀀스 x 시작 부분에 영구 기억을 연결하여 새로운 입력 시퀀스 구성

  (2) 새로 구성된 입력 시퀀스를 SWA 모듈에 입력하여 단기 기억을 나타내는 출력 y 생성

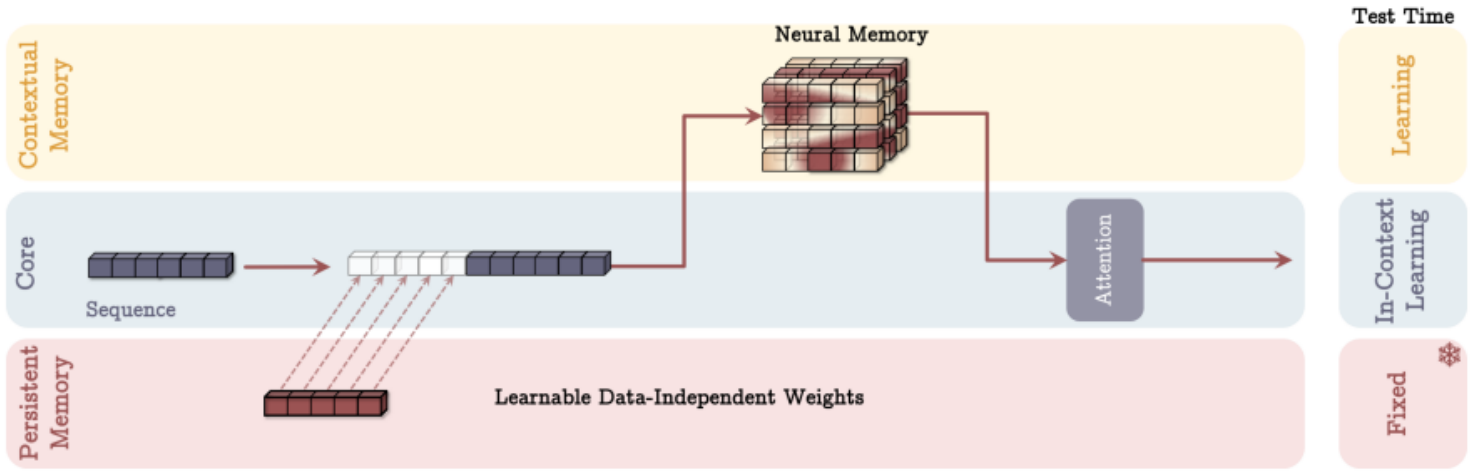
  (3) 동시에 새로 구성된 입력 시퀀스를 LMM에 입력하여 장기 기억을 나타내는 출력 $\mathcal{M}(\tilde{x})$ 생성

  (4) y와 $\mathcal{M}(\tilde{x})$을 비선형 게이트를 통해 결합하여 최종 출력 생성

- 장점: SWA는 정밀한 단기 기억 역할을 하고, LMM은 fading memory(점점 잊혀지는) 역할을 하여 두 가지 다른 유형의 기억을 효과적으로 통합

V Titans Architecture

❖ MAL (Memory As a Layer)



$$\tilde{x} = [p_1 \quad p_2 \quad \cdots \quad p_{N_p}] \;\|\; x,$$
$$y = \mathcal{M}(\tilde{x}),$$
$$o = \texttt{SW-Attn}\,(y),$$

- 장기 기억 모듈을 신경망의 한 layer처럼 배치하고, 그 출력 값을 다시 단기 기억 모듈의 입력으로 사용하는 순차적인 방식
- 이 LMM 레이어는 과거 및 현재 컨텍스트를 압축하는 역할

  (1) 입력 시퀀스 x 시작 부분에 영구 기억을 연결하여 새로운 입력 시퀀스 구성

  (2) 새로 구성된 입력 시퀀스를 LMM 모듈에 입력하여 출력 생성

  (3) LMM의 출력을 SWA에 입력하여 최종 출력 생성

- 기존 모델들이 순환 모델과 Attention을 순차적으로 쌓는 방식과 유사하여 구현이 직관적임

VI Experiments

❖ 언어 모델링 및 상식 추론 성능

- 비하이브리드 모델: 순수 attention/순환 기반 모델
- 하이브리드 모델(*로 표시): attention + 순환 결합 모델
- Titans 변형: 순수 LMM, MAC, MAG, MAL

- 비하이브리드 모델/하이브리드 모델 내에서 Titans의 성능이 가장 우수
- Titans 변형 내에서는 MAL이 다른 두 개에 비해 성능이 떨어짐. 이는 LMM과 attention의 상호 보완적인 정보 처리 능력을 충분히 활용하지 못하기 때문이라고 분석

- 파라미터 수가 증가함에 따라 모든 모델(특히 Titans)의 성능이 향상

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

VI Experiments

❖ NIAH: Needle in a Haystack, 건초더미 속 바늘 찾기

- 매우 긴 컨텍스트 내에서 특정 정보를 얼마나 잘 찾아내는가를 측정
- PK(Perturbed Keys)

  : 단일 문서 안에서 여러 개 바늘이 등장하고, key(검색어)를 살짝 변형 -> 단순 매칭이 아닌 의미 기반 검색을 요구
- N(Needles)

  : 바늘의 개수에 따른 성능 측정 -> 다중 정보 저장 능력을 평가
- W(Window size)

  : 바늘이 위치할 수 있는 window 크기를 다양하게 바꿔가며 성능 측정 -> 긴 문맥에서 위치 의존성을 평가

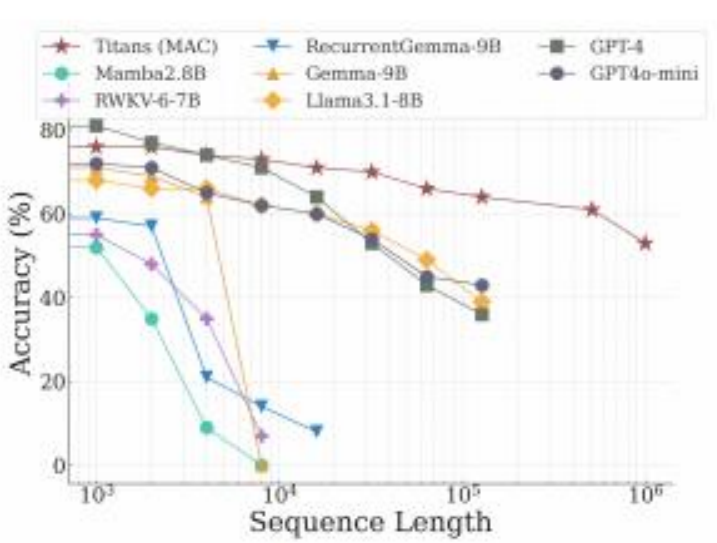
Model	S-NIAH-PK				S-NIAH-N				S-NIAH-W			
	2K	4K	8K	16K	2K	4K	8K	16K	2K	4K	8K	16K
TTT	98.4	98.8	98.0	88.4	60.2	36.6	10.2	4.4	78.8	28.0	4.4	0.0
Mamba2	98.6	61.4	31.0	5.4	98.4	55.8	14.2	0.0	42.2	4.2	0.0	0.0
DeltaNet	96.8	98.8	98.6	71.4	47.2	15.4	12.8	5.4	46.2	20.0	1.6	0.0
Titans (LMM)	99.8	98.4	98.2	96.2	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

VI Experiments

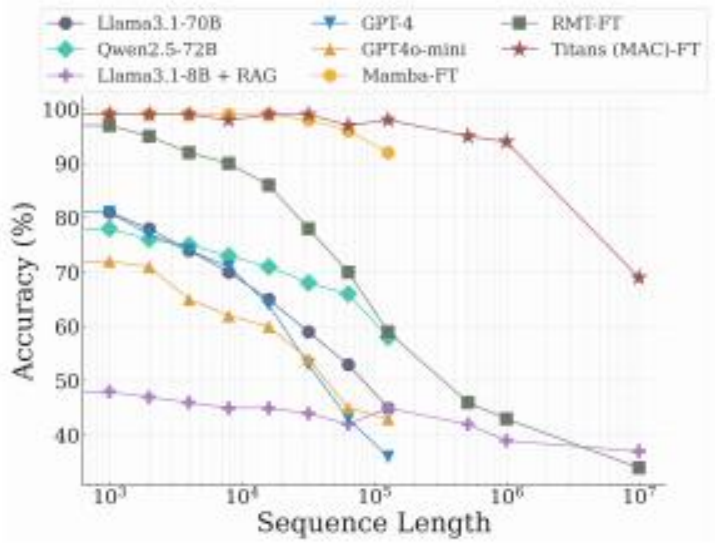
❖ BABILong Benchmark: 초장문 컨텍스트 내 추론 능력을 평가

- NIAH가 단순히 특정 정보를 검색하는 것이었다면, BABILong은 매우 긴 문서에 분산된 여러 사실들을 종합하여 추론하는 능력을 측정

- Few-shot Setup: 사전 학습된 LLM들이 추가적인 fine-tuning 없이도 초장문 컨텍스트에서 얼마나 잘 추론하는지를 평가
  -> 모델의 내재적인 장문 컨텍스트 이해 및 추론 능력을 보여줌
- Fine-tuning Setup: LLM 모델들을 fine-tuning하여 해당 태스크에 얼마나 잘 적응하고 최적화될 수 있는지를 평가
  -> 모델의 학습 능력과 장문 컨텍스트 추론을 위한 적응성을 보여줌



(a) Few-shot Setup



(b) Fine-Tuning Setup

VI Experiments

❖ 기타 실험 결과

- 뿐만 아니라 Deep Memory의 효과, 시계열 예측, DNA 모델링, 효율성, Ablation(제거 또는 대체) 연구에서도 좋은 성능을 보임

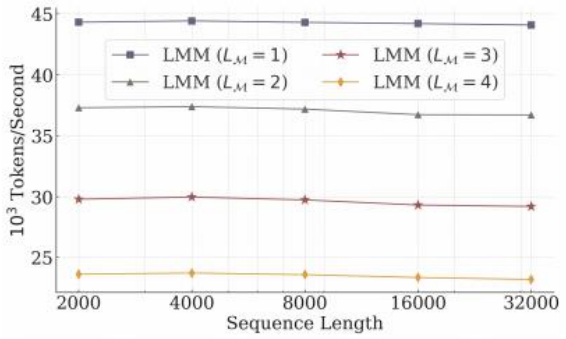


Figure 8: The effect of memory depth on training throughput

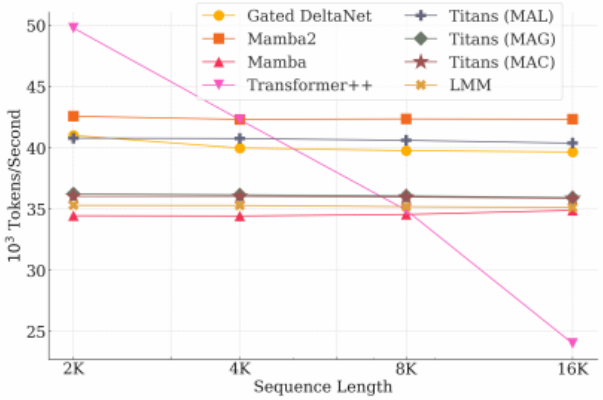


Figure 9: Training throughput comparison of Titans and baselines.

Table 4: Downstream evaluation of pre-trained DNA models on GenomicsBenchmarks (Grešová et al. 2023). We report top-1 classification accuracy (%).

Model	Enhancer Cohn	Enhancer Ens	Human Reg.	Non-TATA Promoters	Human OCR Ens.
CNN	69.5	68.9	93.3	84.6	68.0
DNABERT	74.0	85.7	88.1	85.6	75.1
GPT	70.5	83.5	91.5	87.7	73.0
HyenaDNA	74.2	89.2	93.8	96.6	80.9
Transformer++	73.4	89.5	89.9	94.4	79.5
Mamba	73.0	-	-	96.6	-
Based	74.6	89.5	89.5	96.8	79.0
Neural Memory Module	75.2	89.6	89.3	96.6	79.9

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