Test-Time Training (TTT)

Learning to (Learn at Test Time): RNNs with Expressive Hidden States

Yu Sun[*1], Xinhao Li[*2], Karan Dalal[*3], Jiarui Xu[2], Arjun Vikram[1], Genghan Zhang[1], Yann Dubois[1], Xinlei Chen[†4], Xiaolong Wang[†2], Sanmi Koyejo[†1], Tatsunori Hashimoto[†1], Carlos Guestrin[†1]

[*] Core contributors. [†] Joint advising. See author contributions at the end of the paper.
[1] Stanford University. [2] UC San Diego. [3] UC Berkeley. [4] Meta AI.
Correspondence to: `yusun@cs.stanford.edu`, `xil202@ucsd.edu`, `kdalal@berkeley.edu`.
Code available in JAX and PyTorch.

Motivation for Test-Time Training

- Self-attention受限于$O(n^2)$的二次计算复杂度，其推理延迟与显存占用随序列长度呈非线性增长。

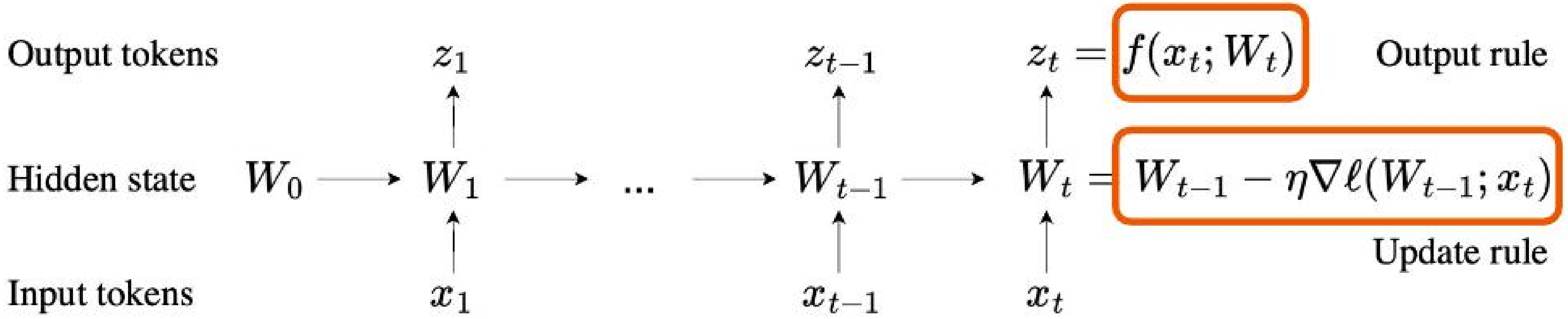
- RNN 架构通过将历史上下文显式压缩至固定维度的隐状态中，实现了 $O(1)$ 的单步推理复杂度和线性级扩展性速度快且高效，但在长上下文任务中缺乏表达能力。

- 挑战在于压缩长序列又不丢失上下文。

- 模型需要在复杂度和表达能力之间取得平衡。

Overview of Test-time Training

- **核心想法:** 模型的权重W设置为隐藏状态，更新隐藏状态进行长文本记忆
- **更新规则:** a gradient step on the self-supervised loss ℓ
- **辅助任务**：在没有标签的测试阶段，创造一个“学习目标” ℓ，以便让模型算出梯度来更新隐藏状态



如何设计辅助任务

- 辅助任务被设计为：**重构**和**去噪**

- 模型并不直接看到完整的**输入 Token x_t**，而是看到一个被“破坏”或“降维”后的版本，称为**部分信息 $\tilde{x}_t$**。模型f(也就是TTT层)的任务是利用残缺的$\tilde{x}_t$去恢复/猜测出原始的 x_t。

$$l(W; x_t) = ||f(\tilde{x}_t; W) - x_t||^2$$

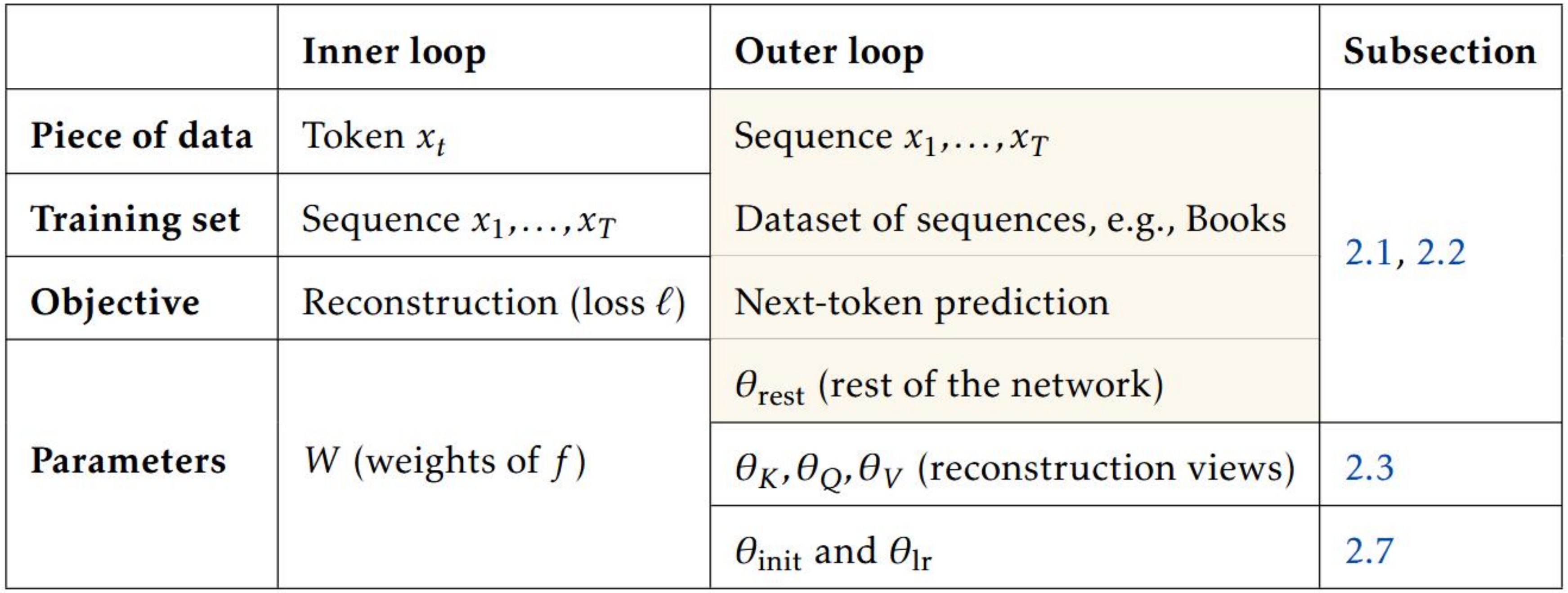
- 用这个Loss对权重W求导，进行梯度下降

Key-value Binding (TTT-KVB)

- 从“重建x”到绑定“K, V”
  - 模仿Transformer的关联存储，引入θ_K和θ_V，使得权重W可以捕捉到更为复杂的特征，增强自监督任务的“隐状态表现力”。

$$\ell(W; x_t) = \left\| f\left(\theta_K x_t; W\right) - \theta_V x_t \right\|^2 .$$

TTT总结

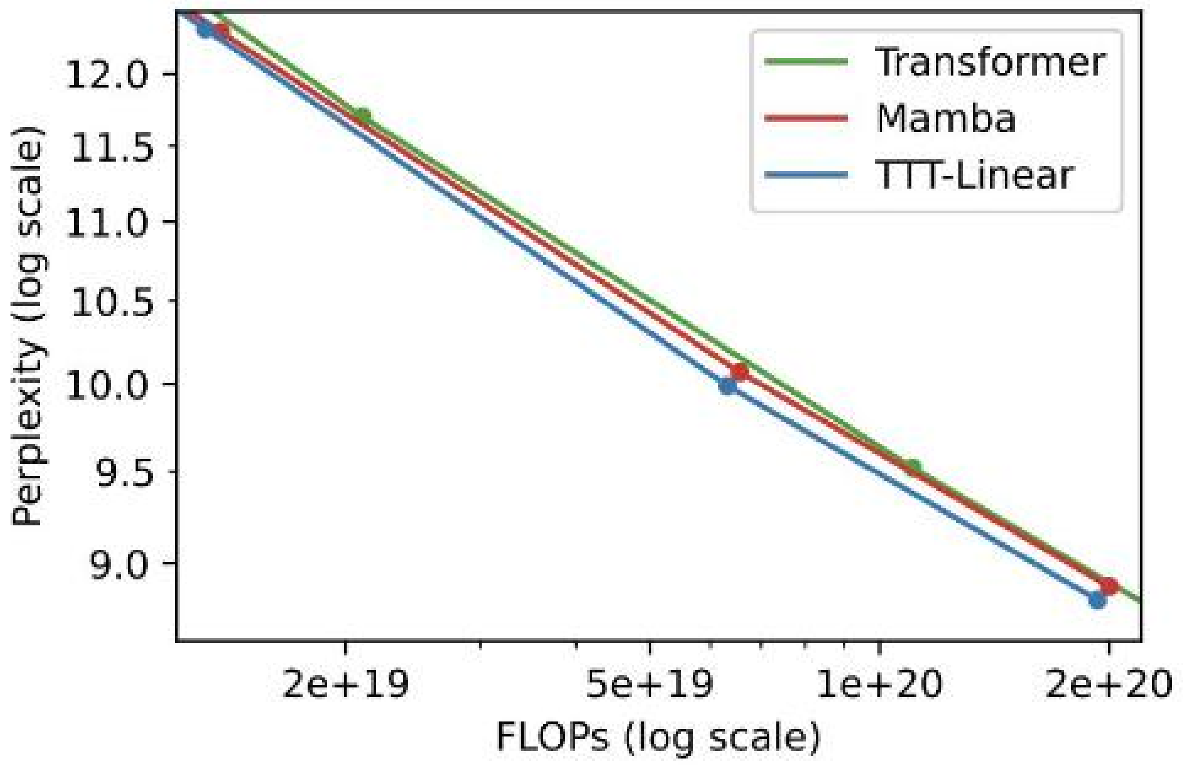


	Inner loop	**Outer loop**	**Subsection**
Piece of data	Token x_t	Sequence $x_1, \ldots, x_T$	2.1, 2.2
Training set	Sequence $x_1, \ldots, x_T$	Dataset of sequences, e.g., Books	
Objective	Reconstruction (loss ℓ)	Next-token prediction	
Parameters	W (weights of f)	θ_{rest} (rest of the network)	
		$\theta_K, \theta_Q, \theta_V$ (reconstruction views)	2.3
		θ_{init} and θ_{lr}	2.7

Table 2. In summary, our paper reformulates supervised learning as learning to learn, with two nested loops. Highlighted rows of the outer loop are the same as in the regular training. Parameters of the outer loop become hyper-parameters of the inner loop. Intuitively, the inner loop, *i.e.* TTT, is "one level below" regular training.

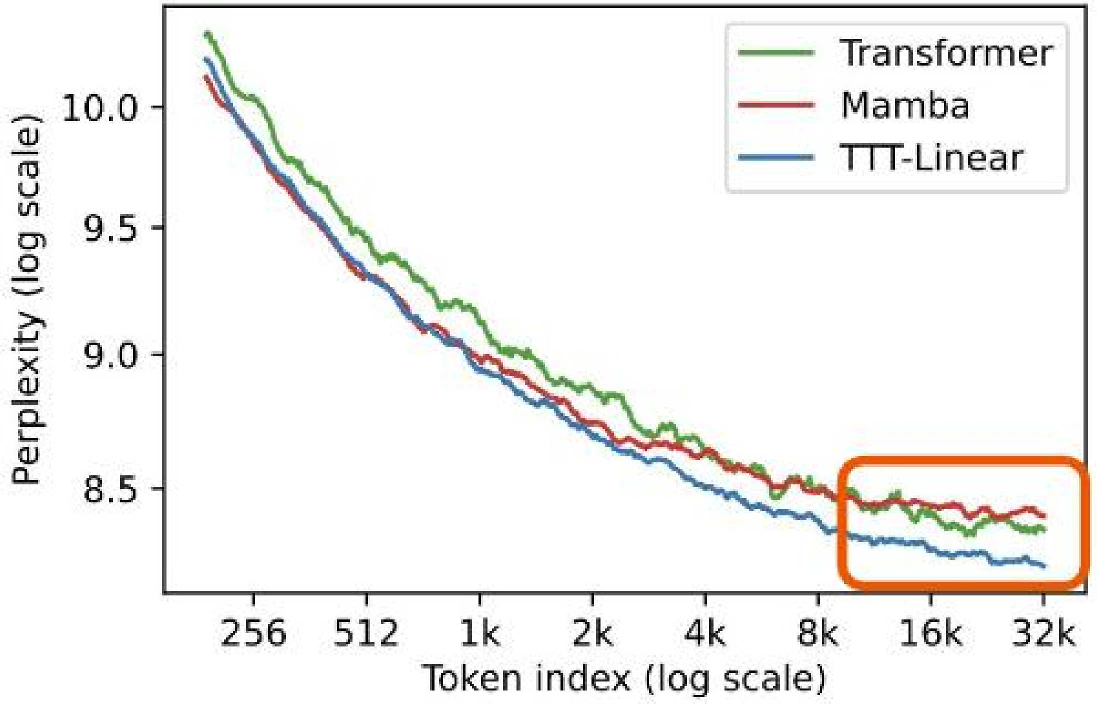
Better perplexity and Fewer FLOPs

- 通过mini-batch和对偶形式等技巧提高 TTT 层的硬件效率，TTT-Linear可以以更小的FLOPs达到同样的困惑度。

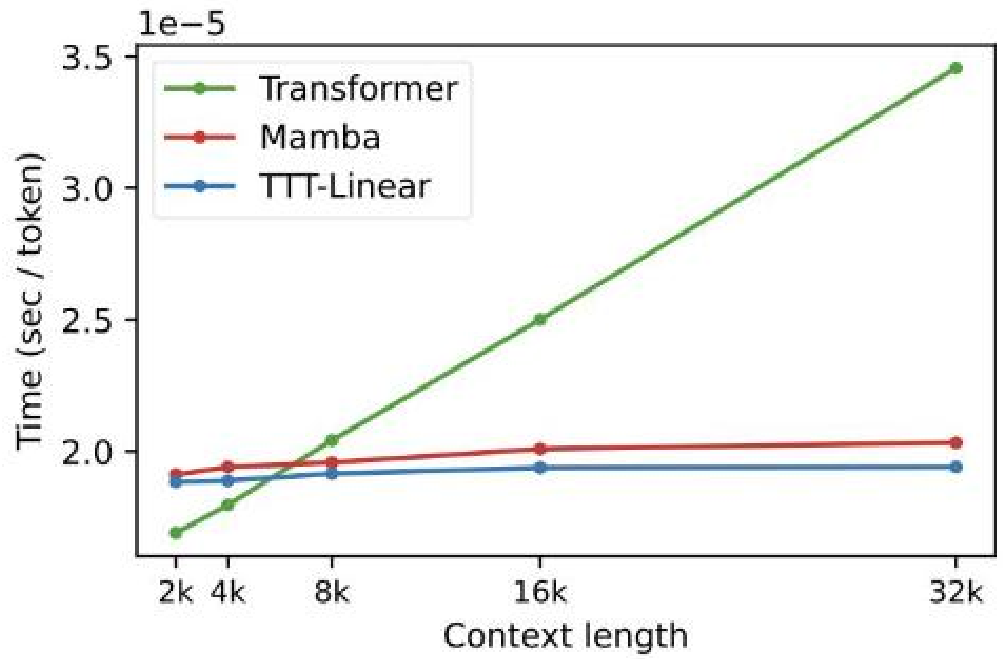


Better use of long context

- 与Transformer类似，TTT-Linear可以通过以更多标记为条件来不断降低困惑度，而Mamba在上下文达到16k后就无法做到这一点了。



Constant Forward Time per tokens



- 随着上下文长度的增加，Transformer的每个token前向时间呈线性增长，但其他两种方法的每个token前向时间大致保持不变。

- 在8k上下文之后，TTT-Linear比Transformer更快，且与Mamba相当。

Beyond 32K

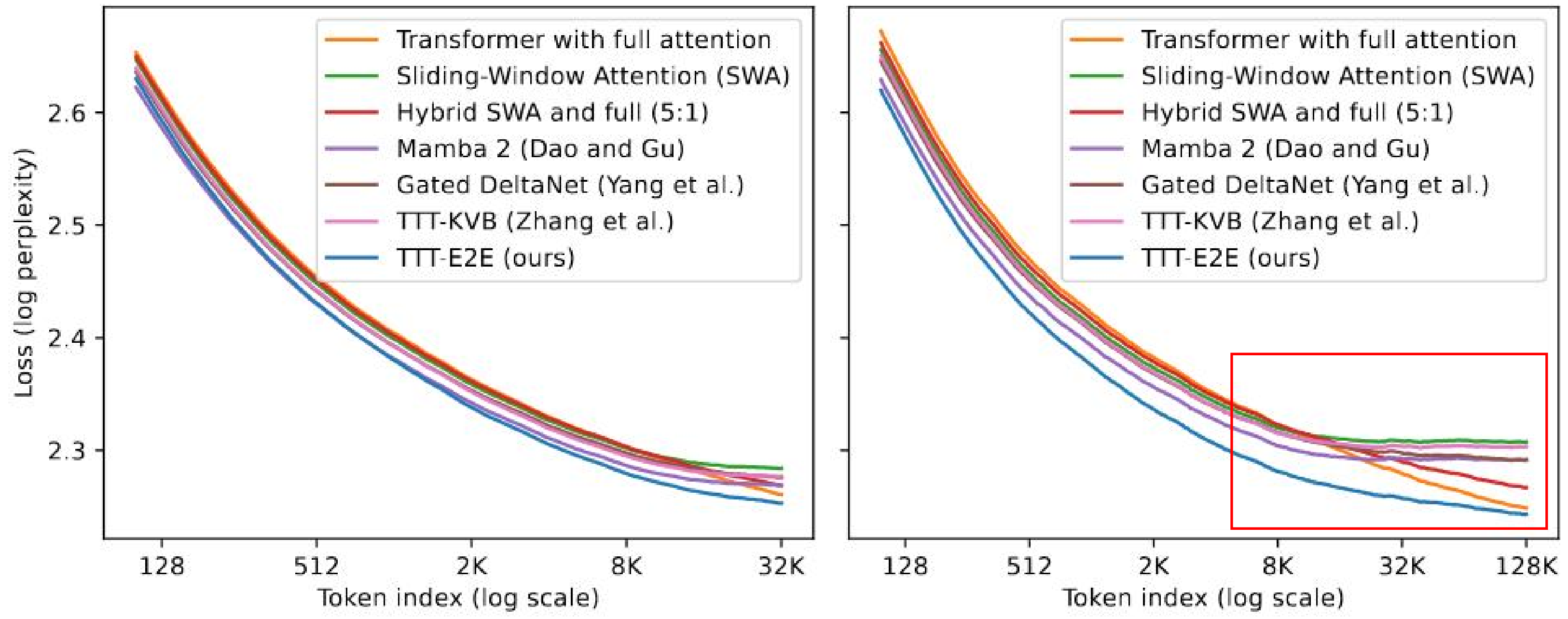


Figure 6. Loss breakdown by token index, for context length 32K (left) and 128K (right), following the same process as when we produced the right panel of Figure 2; see details in Subsection 3.4. Overall, TTT-E2E is the only method that always achieves lower losses than full attention throughout the entire context length, and its aggregated advantage mostly comes from the earlier tokens.

Limitations

- 非端到端优化目标，导致训练和测试时优化目标不一致，使得长序列建模能力降低。
- 每一层都引入新的训练参数θ_K和θ_V，导致计算量增大
- 当处理的token量过大时（超过32K），Loss会反弹，长文本建模能力下降
- 考虑到计算效率，隐藏状态较小（用LoRA实现），限制了隐藏状态的存储能力

End-to-End Test-Time Training for Long Context

Arnuv Tandon*[1,3], Karan Dalal*[1,4], Xinhao Li*[5], Daniel Koceja*[3], Marcel Rød*[3], Sam Buchanan[4], Xiaolong Wang[5], Jure Leskovec[3], Sanmi Koyejo[3], Tatsunori Hashimoto[3], Carlos Guestrin[3], Jed McCaleb[1], Yejin Choi[2], Yu Sun*[2,3]

[1] Astera Institute [2] NVIDIA [3] Stanford University [4] UC Berkeley [5] UC San Diego

端到端

- 去掉了每一层的“辅助任务”
- 改为Next-Token Prediction，实现了端到端
- 将训练时的优化目标和测试时的最终目标统一
- 将长文本建模构建为“持续学习”问题（打通了训练和测试界限）

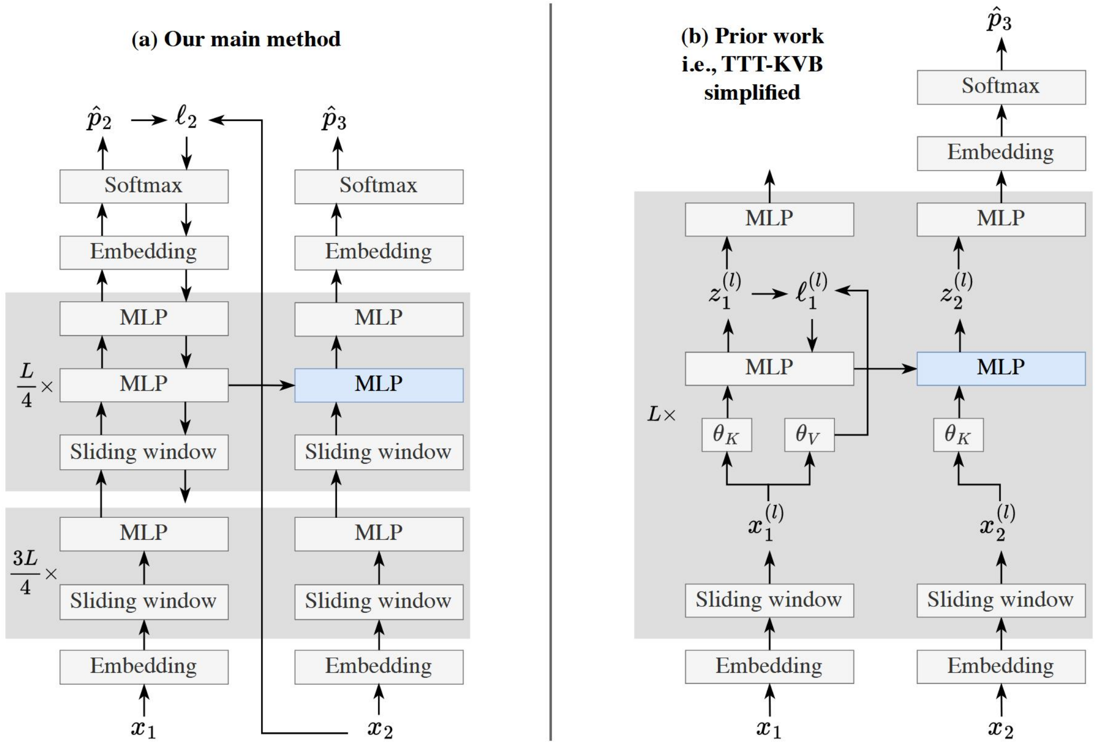


Figure 3. Computation graphs following the setup in Figure 2: Given x_1 and x_2 as context, we want to predict the unknown x_3. **Left:** Our main method with the sliding-window attention layers and the implementation details discussed in Subsection 2.3. For ease of notation, our illustration uses online gradient descent ($b = 1$). The lowest downward arrow is disconnected to the MLP below, since gradients pass through the last $L/4$ blocks but not further down. **Right:** The first step of our alternative derivation in Subsection 2.4: a simplified version of TTT-KVB in prior work [110, 87].

架构分工：长短期记忆的协同

- 局部上下文（短期记忆）：由滑动窗口注意力负责。SWA存在于每一层，已经处理好了最近的几千个token的信息。
- 全局上下文（长期记忆）：由TTT更新的动态层负责。
- 双层MLP，冻结其中一个，更新另一个，增加一个静态的、额外的MLP层，用于存储预训练知识，防止遗忘。

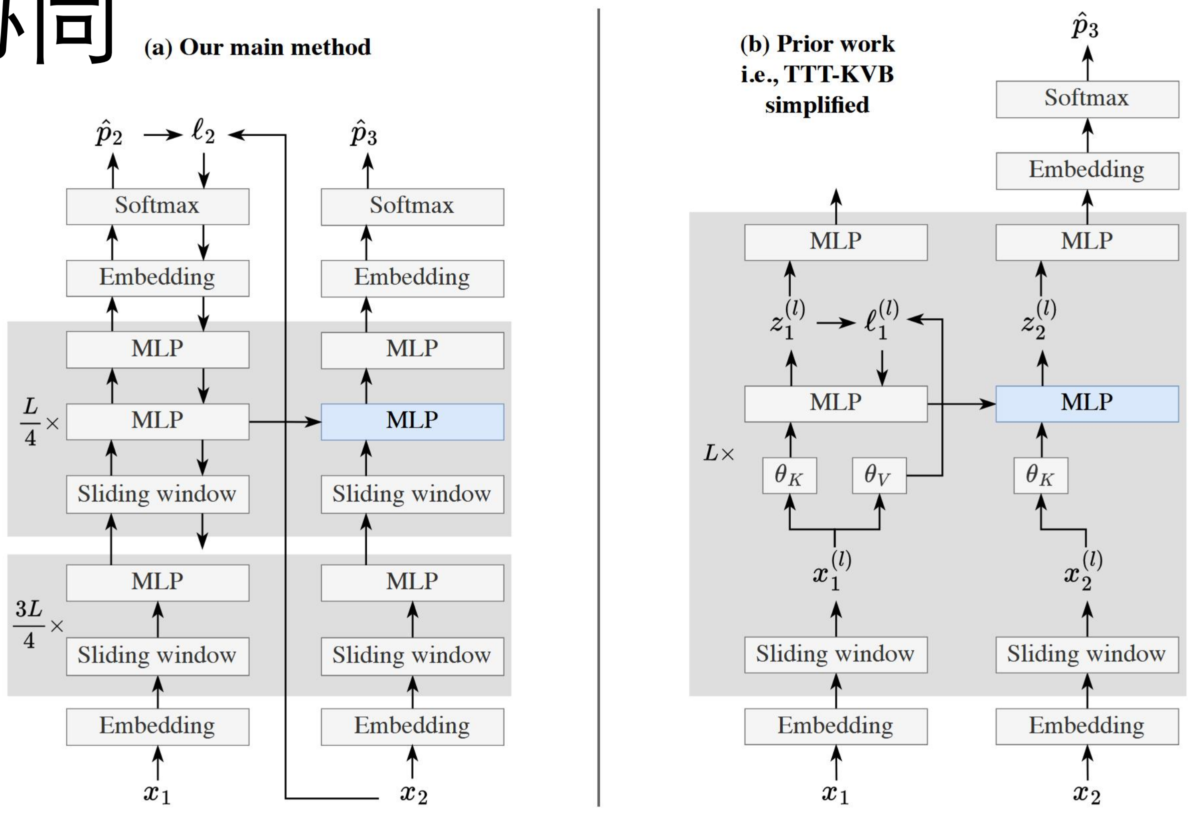


Figure 3. Computation graphs following the setup in Figure 2: Given x_1 and x_2 as context, we want to predict the unknown x_3. **Left:** Our main method with the sliding-window attention layers and the implementation details discussed in Subsection 2.3. For ease of notation, our illustration uses online gradient descent ($b = 1$). The lowest downward arrow is disconnected to the MLP below, since gradients pass through the last $L/4$ blocks but not further down. **Right:** The first step of our alternative derivation in Subsection 2.4: a simplified version of TTT-KVB in prior work [110, 87].

引入元学习

- **为什么需要元学习出最佳初始参数W_0**：在传统的动态评估（Dynamic Evaluation）中，模型是在标准预训练下优化 W 的，并没有考虑到 W 会在测试时被进一步修改 。

- **元学习的解决方案：**通过外循环优化W_0，模型在训练时学会了在测试时看到新上下文并进行梯度更新时表现较好。

模拟“test-time training”

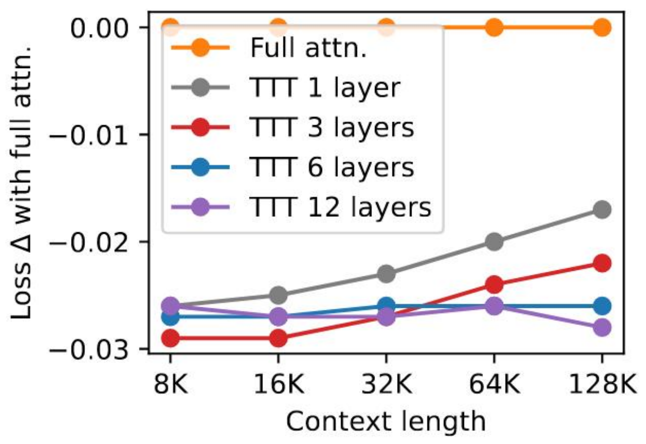
1. 输入序列：给模型一段长文本
2. 内循环演练：模型从当前的W_0出发。每读到一个mini-batch的数据，就根据公式计算梯度，并临时更新权重，得到$W_1, W_2, \dots$。
3. 评估表现：看看这些更新后的权重（如W_{i-1}）在预测下一个batch的token时准不准。

计算“元梯度”，并在外循环更新

1. 如果模型更新后的权重预测得不准，系统会产生一个损失
2. 由于W_{i-1}是从W_0经过梯度下降计算得到，可以用反向传播计算出梯度的梯度，更新W_0

仅更新部分网络

- 由于推理时，每处理一个batch都要进行一次反向传播来更新权重。（尤其是涉及元学习，需要计算梯度的梯度）
- 如果更新全网络的所有参数，推理速度会显著变慢，只更新每个Block中1/4的MLP层可以同时兼顾极高性能和降低计算量



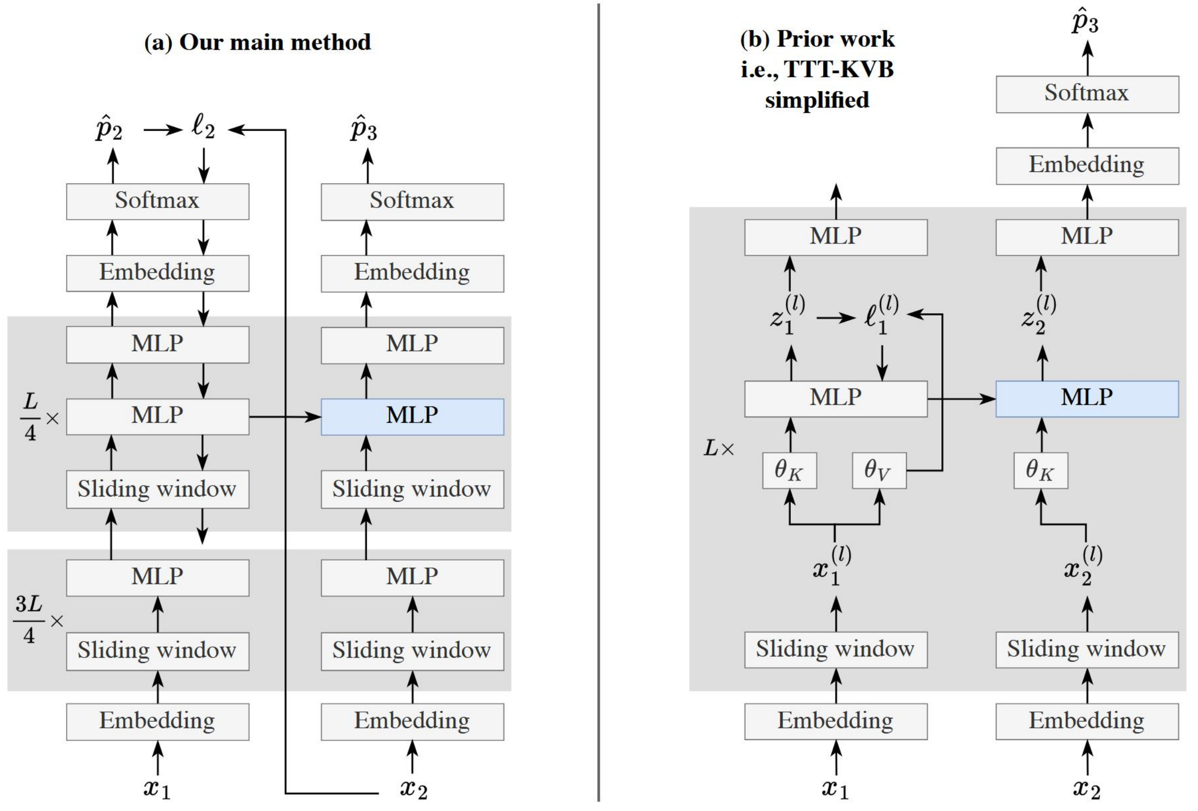
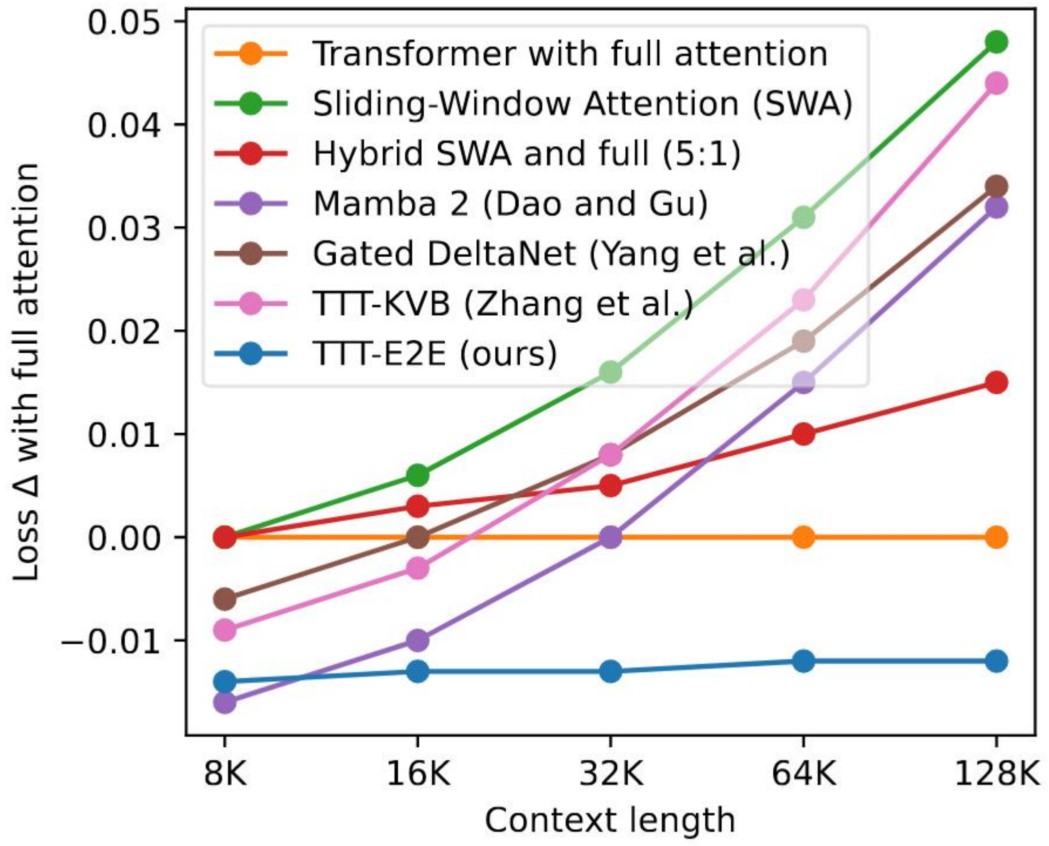
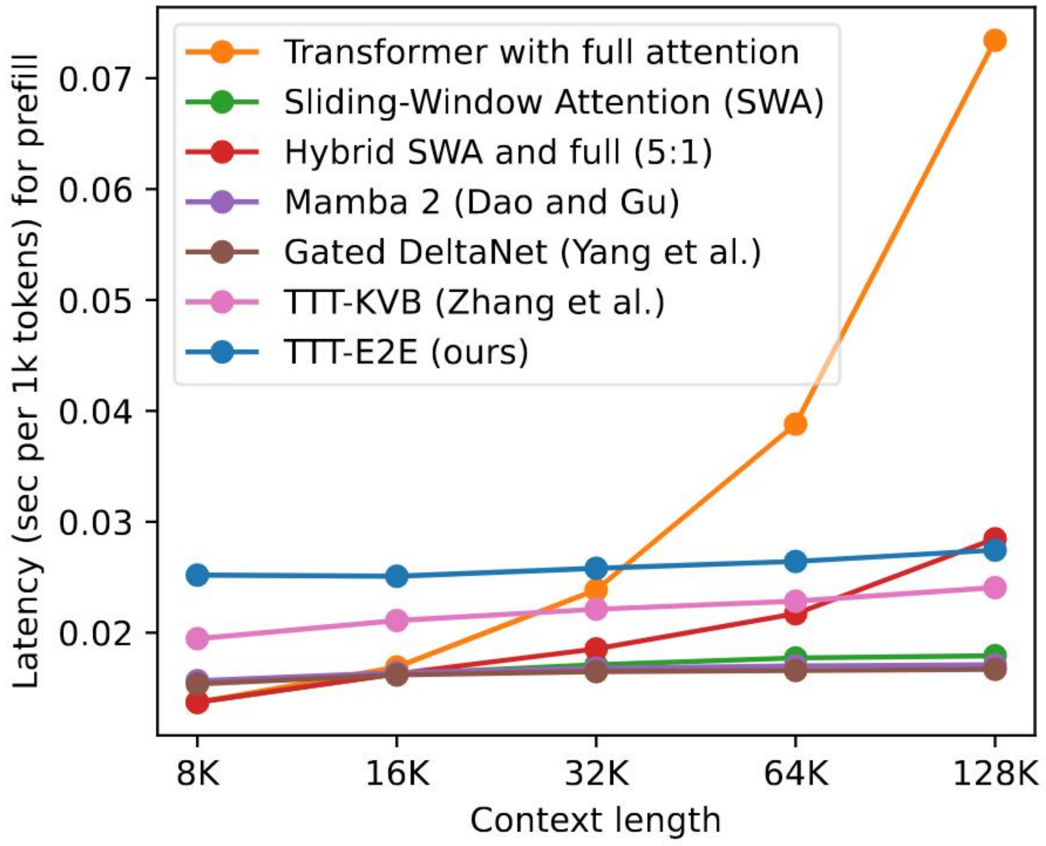


Figure 3. Computation graphs following the setup in Figure 2: Given x_1 and x_2 as context, we want to predict the unknown x_3. **Left:** Our main method with the sliding-window attention layers and the implementation details discussed in Subsection 2.3. For ease of notation, our illustration uses online gradient descent ($b = 1$). The lowest downward arrow is disconnected to the MLP below, since gradients pass through the last $L/4$ blocks but not further down. **Right:** The first step of our alternative derivation in Subsection 2.4: a simplified version of TTT-KVB in prior work [110, 87].

在保持RNN级恒定速度的同时，实现全注意力机制的性能扩展

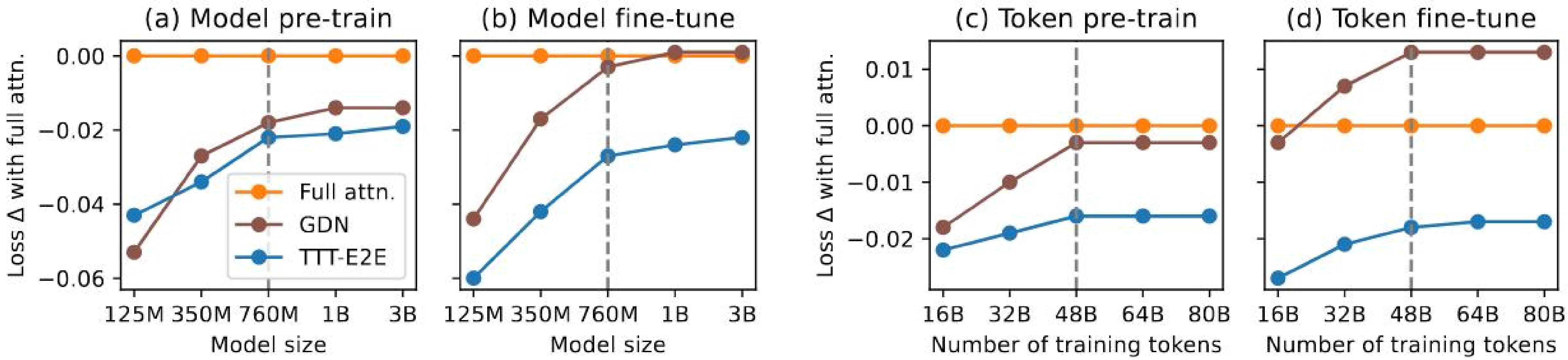
$$\text{Loss }\Delta = \text{Loss}_{\text{i}} - \text{Loss}_{\text{full } attn}$$





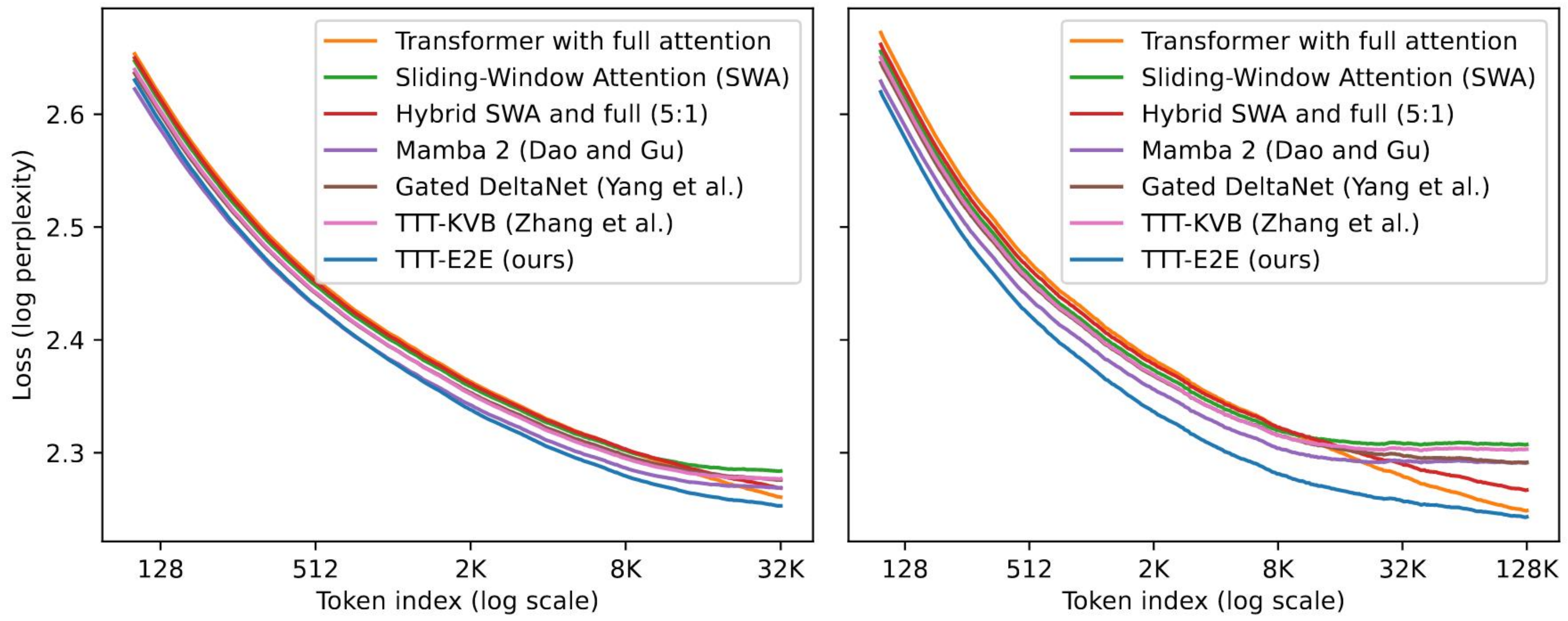
稳健性：

$$\text{Loss}\,\Delta \;=\; \text{Loss}_{\text{i}} - \text{Loss}_{\text{full }attn}$$



- 虽然没有展示规模更大时候的效果，但是趋势显示，当继续scale up，loss会依旧低于full attention
- 当训练token数不断拉高，loss也会保持平缓。

TTT-E2E与baseline的比较



消融实验：

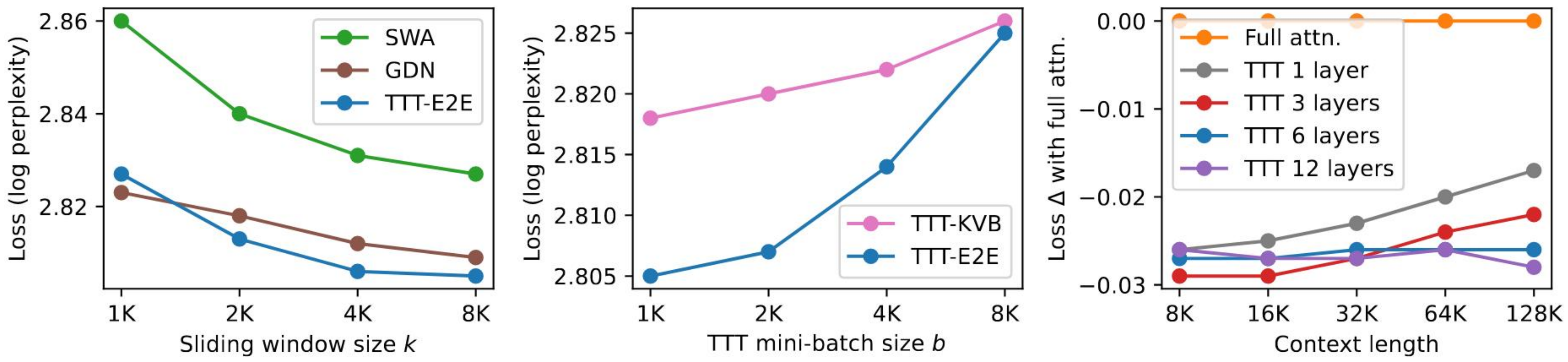


Figure 4. Ablations on three hyper-parameters: sliding window size k, mini-batch size b, and the number of layers updated during TTT; see details in Subsection 3.2. Given the trends in these ablations, we set $k = 8\text{K}$, $b = 1\text{K}$, and we update 1/4 the total number of layers. Loss Δ (↓), the y-value in the rightmost panel, is the same as in Figure 1. It is computed as (loss of the reported method) − (loss of Transformer with full attention), so loss Δ of full attention itself (orange) is the flat line at $y = 0$. GDN stands for Gated DeltaNet [105].

Limitations

- 训练延迟较高，训练时间极长，换取较快推理时间

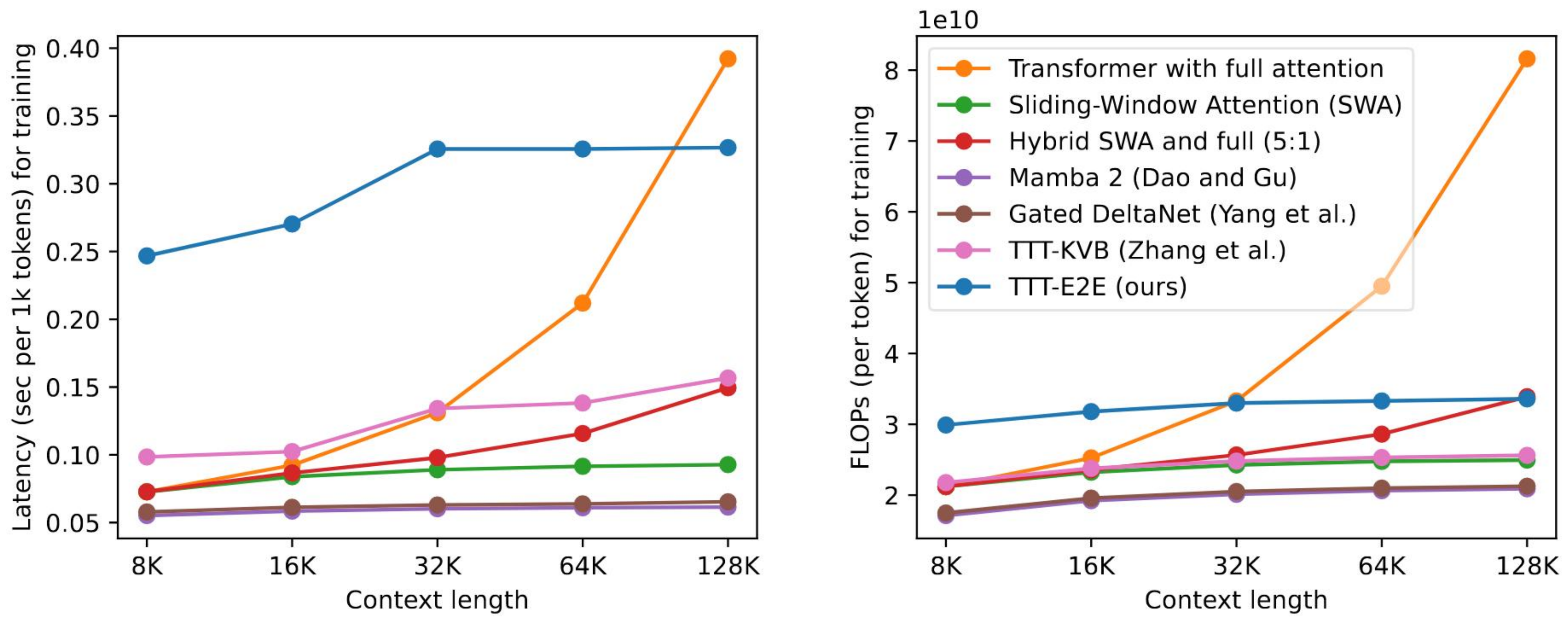
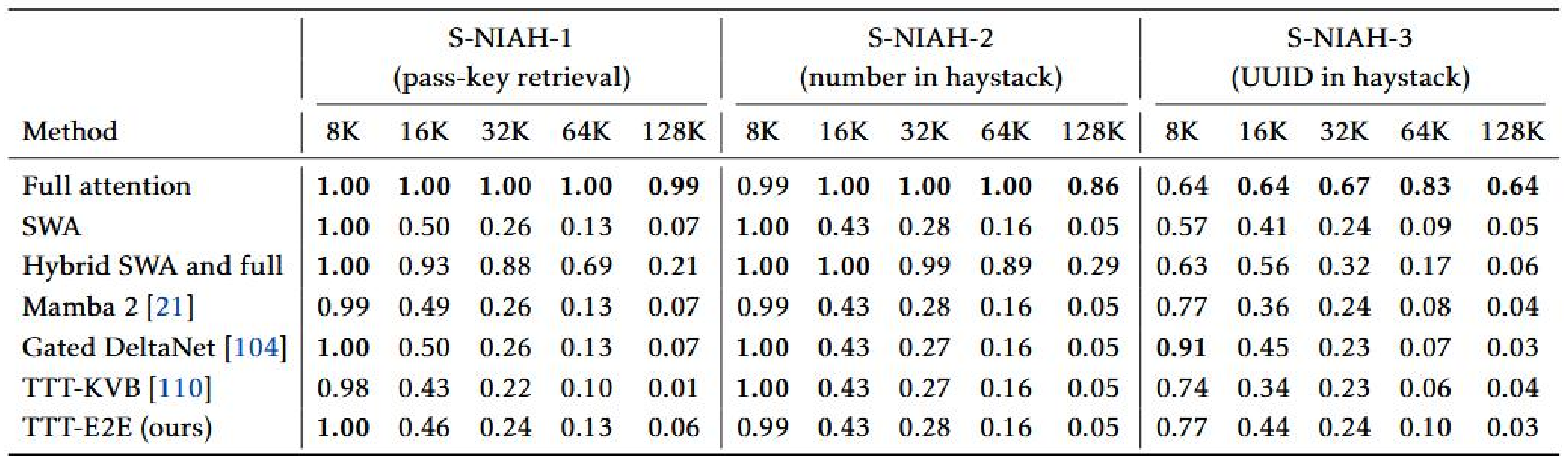


Figure 8. Training efficiency, in terms of latency on an H200 (left) and FLOPs (right); see details in Subsection 3.3. Overall, training latency is still a significant limitation of our current implementation. The legend is shared across both panels.

Limitations

- 本质上还是有损压缩，全线不如Full attention



	S-NIAH-1 (pass-key retrieval)					S-NIAH-2 (number in haystack)					S-NIAH-3 (UUID in haystack)				
Method	8K	16K	32K	64K	128K	8K	16K	32K	64K	128K	8K	16K	32K	64K	128K
Full attention	**1.00**	**1.00**	**1.00**	**1.00**	**0.99**	0.99	**1.00**	**1.00**	**1.00**	**0.86**	0.64	**0.64**	**0.67**	**0.83**	**0.64**
SWA	**1.00**	0.50	0.26	0.13	0.07	**1.00**	0.43	0.28	0.16	0.05	0.57	0.41	0.24	0.09	0.05
Hybrid SWA and full	**1.00**	0.93	0.88	0.69	0.21	**1.00**	**1.00**	0.99	0.89	0.29	0.63	0.56	0.32	0.17	0.06
Mamba 2 [21]	0.99	0.49	0.26	0.13	0.07	0.99	0.43	0.28	0.16	0.05	0.77	0.36	0.24	0.08	0.04
Gated DeltaNet [104]	**1.00**	0.50	0.26	0.13	0.07	**1.00**	0.43	0.27	0.16	0.05	**0.91**	0.45	0.23	0.07	0.03
TTT-KVB [110]	0.98	0.43	0.22	0.10	0.01	**1.00**	0.43	0.27	0.16	0.05	0.74	0.34	0.23	0.06	0.04
TTT-E2E (ours)	**1.00**	0.46	0.24	0.13	0.06	0.99	0.43	0.28	0.16	0.05	0.77	0.44	0.24	0.10	0.03

Table 2. S-NIAH performance across context lengths, with the best results in bold; see details in Subsection 3.5. Overall, Transformer with full attention dramatically outperforms the other methods, including ours, especially in long context. This observation, combined with findings from our previous subsections, supports the intuition that the strength of full attention lies in its nearly lossless recall.