Forgetting Transformer: Softmax Attention with a Forget Gate

***Zhixuan Lin*[1], *Evgenii Nikishin*[1], *Xu Owen He*[2], *Aaron Courville*[1]**

[1]Mila & Université de Montréal
[2]MakerMaker AI

In ICLR 2025

Code available at: https://github.com/zhixuan-lin/forgetting-transformer

2025-03-20

Overview

- Motivation
- Forgetting Attention and **Fo**rgetting **Trans**former (**FoX**)
- FlashAttention-based Implementation
- Results and analyses
- Future directions

Motivation

Motivation

- Forget gates are ubiquitous in **recurrent sequence models** (e.g., linear attention)
  - LSTM, Gated RFA, GLA, Mamba-2, HGRN2, RWKV series, xLSTM, Gated DeltaNet…
- Forget gates are known to be crucial for performance
  - See Greff et al., 2016; Van Der Westhuizen & Lasenby, 2018; Peng et al., 2021; Yang et al., 2023; Gu & Dao, 2023
- **Softmax attention** and **linear attention** are very similar **in form**.
  - Softmax attention can be seen as linear attention with an infinite dimensional state
    - $\exp(q^\top k) = \langle \phi(q), \phi(k) \rangle$, where ϕ maps to an infinite dimensional space
  - So let's try adding a forget gate to softmax attention!

Why are Forget Gates Useful?

- Every model has limited modeling capabilities:
  - For recurrent models: limited by #**params** and state size
  - For Transformers: limited #**params**
- Forgetting makes things easier to model (less things to process)
  - **NOTE:** there are many heads. There could still be heads that DO NOT forget

Method

Intuition

- Forget gates ⇔ data-dependent decay of input-output dependencies
- Consider a minimal RNN mapping $(x_i)_{i=1}^L$ to $(o_i)_{i=1}^L$ (everything in $\mathbb{R}^d$)
  - $h_t = f_t h_{t-1} + x_t$, where $f_t \in (0,1)$
  - $o_t = h_t$
- Easy to show: $o_i = \sum_{i=1}^{j} F_{ij} x_j$, where $F_{ij} = \prod_{l=j+1}^{i} f_l$

Takeaway: models without (explicit) recurrence can also have a forget gate

Forgetting Attention

- Softmax attention with a forget gate
  - $q_i, k_i, v_i = W_q x_i, W_k x_i, W_v x_i \in \mathbb{R}^d$
  - $f_i = \sigma(w_f x_i + b_f) \in \mathbb{R}$
  - $F_{ij} = \prod_{l=j+1}^{i} f_l$
  - $o_i = \dfrac{\sum_{j=1}^{i} F_{ij} \exp(q_i^\top k_j) v_j}{\sum_{j=1}^{i} F_{ij} \exp(q_i^\top k_j)}$
- Logit bias form
  - $o_i = \dfrac{\sum_{j=1}^{i} \exp(q_i^\top k_j + d_{ij}) v_j}{\sum_{j=1}^{i} \exp(q_i^\top k_j + d_{ij})}$
  - where $d_{ij} = \log F_{ij} = \sum_{l=j+1}^{i} \log f_l$
- Matrix form:
  - $D = \log F$
  - $O = \mathrm{softmax}(QK^\top + D)V$

Comments

- $\frac{1}{\sqrt{d}}$ logit scaling is omitted in the previous slide; in practice we always use it
- For MHA, one forget gate per head: $f_t^{(h)} = \sigma(w_f^{(h)} x_t + b_f^{(h)})$
  - Sharing a forget gate across heads performs poorly.
- Forget gates are scalar-valued. So the additional computation and parameters are negligible.
- No need for RoPE, so
  - Convenient for long context fine-tuning: no need to adjust θ (e.g. in YaRN)
  - Convenient for non-standard attention such as MLA and NSA
  - It is elegant :)
- Hyperparameter-free, unlike RoPE or ALiBi

Connections to Prior/Concurrent Work

- Connection to linear attention models with forget gates (e.g., GLA, Mamba-2)
- Connection to Attention with Linear Bias (ALiBi)
- Connection to stick-breaking attention

Parallel Form of Gated Linear Attention

Parallel form of linear attention

$$o_i = \frac{\sum_{j=1}^{i} (\phi(q_i)^\top \phi(k_j)) v_j}{\sum_{j=1}^{i} \phi(q_i)^\top \phi(k_j)}$$

Recurrent form of linear attention

- $S_t = S_{t-1} + v_t \phi(k_t)^\top$
- $z_t = z_{t-1} + \phi(k_t)$
- $o_t = \dfrac{S_t \phi(q_t)}{z_t^\top \phi(q_t)}$

Add a forget gate $f_t = \sigma(w_f x_t)$

- $S_t = f_t S_{t-1} + v_t \phi(k_t)^\top$
- $z_t = f_t z_{t-1} + \phi(k_t)$
- $o_t = \dfrac{S_t \phi(q_t)}{z_t^\top \phi(q_t)}$

Recurrent form of **gated** linear attention

Shown in GLA and Mamba-2

$$o_i = \frac{\sum_{j=1}^{i} F_{ij} (\phi(q_i)^\top \phi(k_j)) v_j}{\sum_{j=1}^{i} F_{ij} \phi(q_i)^\top \phi(k_j)}$$

where $F_{ij} = \prod_{l=j+1}^{i} f_l$

Parallel form of **gated** linear attention

Gated Linear Attention to Forgetting Attention

$$o_i = \frac{\sum_{j=1}^{i} F_{ij}(\phi(q_i)^\top \phi(k_j))v_j}{\sum_{j=1}^{i} F_{ij}\phi(q_i)^\top \phi(k_j)}$$

$$\xrightarrow{\phi(q_i)^\top \phi(k_j) \to \exp(q_i^\top k_j)}$$

$$o_i = \frac{\sum_{j=1}^{i} F_{ij} \exp(q_i^\top k_j)v_j}{\sum_{j=1}^{i} F_{ij} \exp(q_i^\top k_j)}$$

Connection to ALiBi

- Logit bias form
  - $o_i = \dfrac{\sum_{j=1}^{i} \exp(q_i^\top k_j + d_{ij}) v_j}{\sum_{j=1}^{i} \exp(q_i^\top k_j + d_{ij})}$
  - where $d_{ij} = \log F_{ij} = \sum_{l=j+1}^{i} \log f_l$
- Let $f_l = \exp(-m)$, then $d_{ij} = -m(j-i)$, which is **ALiBi.**
- So: Forgetting Attention can be seen a **learnable** and **data-dependent** version of ALiBi
  - And naturally works much better in practice!

Connection to Stick-Breaking Attention (SBA)

- SBA:
  - $\beta_{ij} = \sigma(q_i^\top k_j)$
  - $A_{i,j} = \boxed{\beta_{i,j}} \boxed{\prod_{j<k<i} (1 - \beta_{i,k})}$
  - $o_i = \sum_{j=1}^{i-1} A_{i,j} v_i$

Retrieval Decay

- Similarity
  - Both have data-dependent decay
- Difference
  - Retrieval and decay
    - $\beta_{i,j}$ in SBA is responsible for both retrieval and decay
    - In Forgetting Attention, softmax is responsible for retrieval, f_t is responsible for decay
  - SBA decay is query-dependent
  - SBA decay is meant for implementing "first match", instead of "doing retrieval within a local scope" (though it can certainly do that)

But you can combine Forgetting Attention and SBA; just add the F_{ij} factor!

Implementation

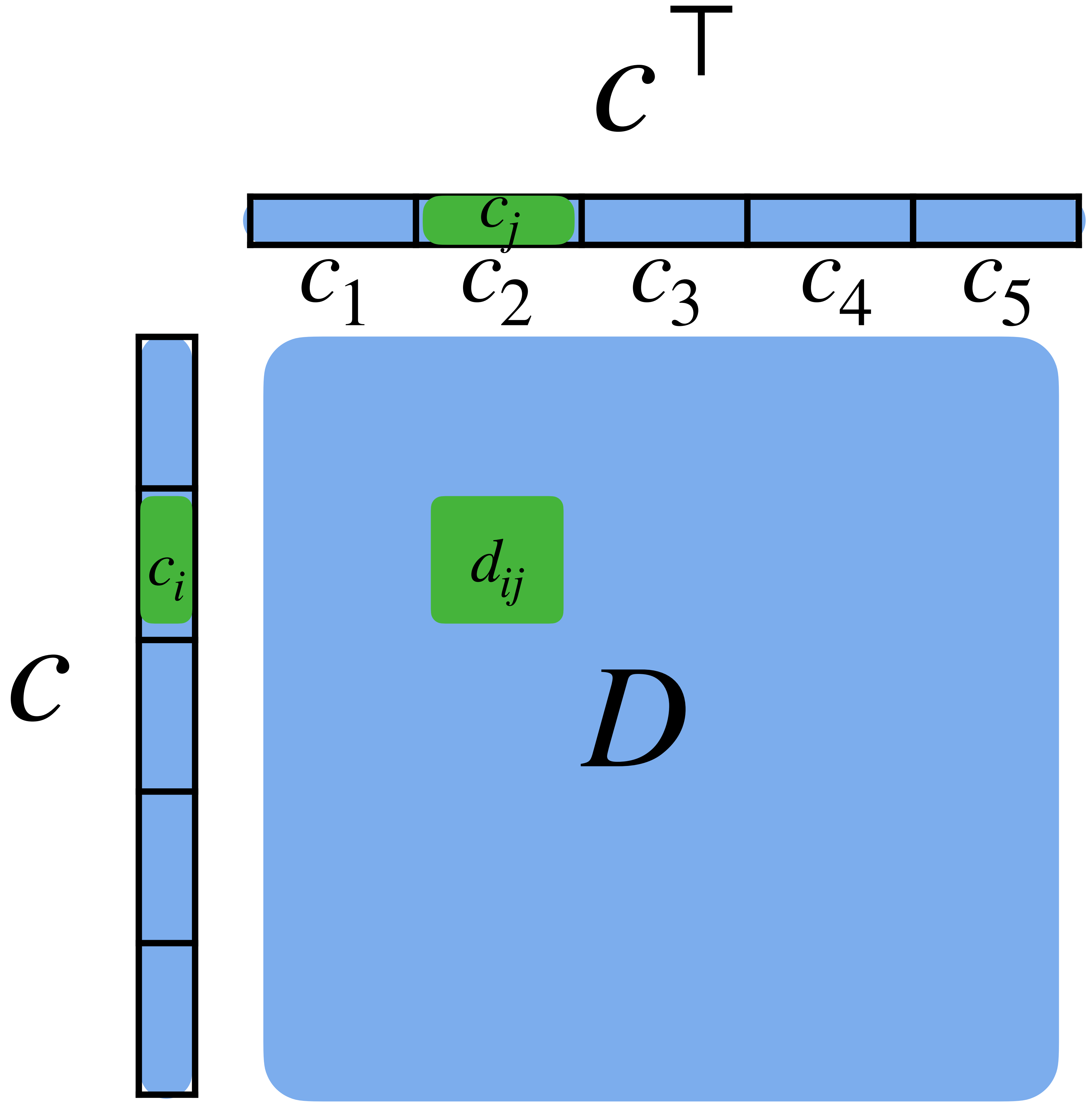
Efficient Implementation

- Logit bias form
  - $o_i = \dfrac{\sum_{j=1}^{i} \exp(q_i^\top k_j + d_{ij}) v_j}{\sum_{j=1}^{i} \exp(q_i^\top k_j + d_{ij})}$
  - where $d_{ij} = \displaystyle\sum_{l=j+1}^{i} \log f_l$
  - $O = \text{softmax}(QK^\top + D)V$

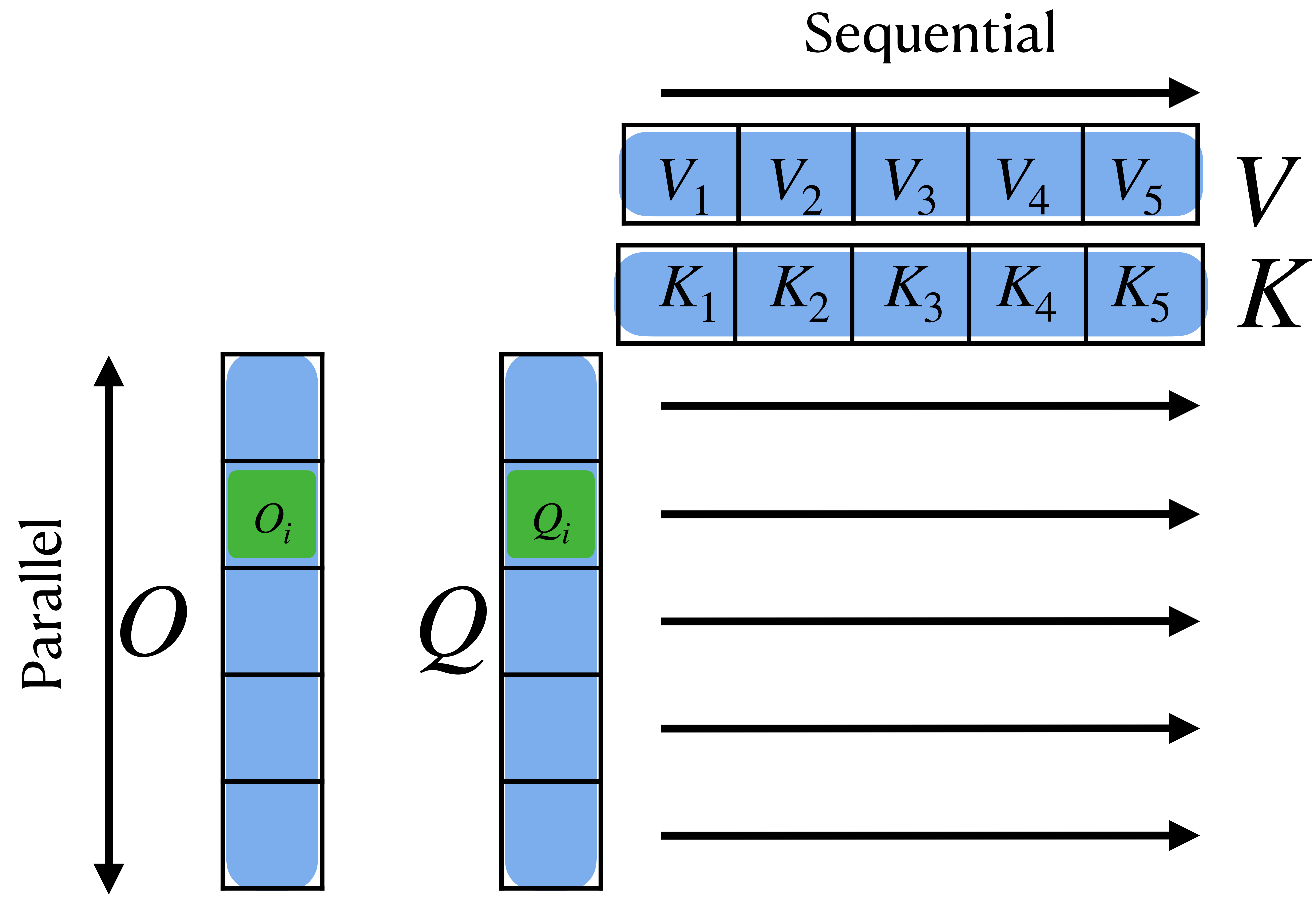
Only need a simple modification to **Flash Attention**!

- Trick for computing $d_{ij} = \displaystyle\sum_{l=j+1}^{i} \log f_l$
  - First compute $c_i = \displaystyle\sum_{l=1}^{i} \log f_l$.
    - $c_1 = \log f_1$,
    - $c_2 = \log f_1 + \log f_2$,
    - ...
    - $c_4 = \log f_1 + \log f_2 + \log f_3 + \log f_4$
  - Then we have $d_{ij} = c_i - c_j$

Efficient Computation of D



Add it to FlashAttention



$$Y_i^{(j)} \leftarrow Y_i^{(j-1)} + \exp(Q_i K_j^\top) V_j$$

$$Z_i^{(j)} = Z_i^{(j-1)} + \exp(Q_i K_j^\top) 1$$

$$D_{ij} = c_i 1^\top - 1 c_j^\top$$

$$Y_i^{(j)} \leftarrow Y_i^{(j-1)} + \exp(Q_i K_j^\top + D_{ij}) V_j$$

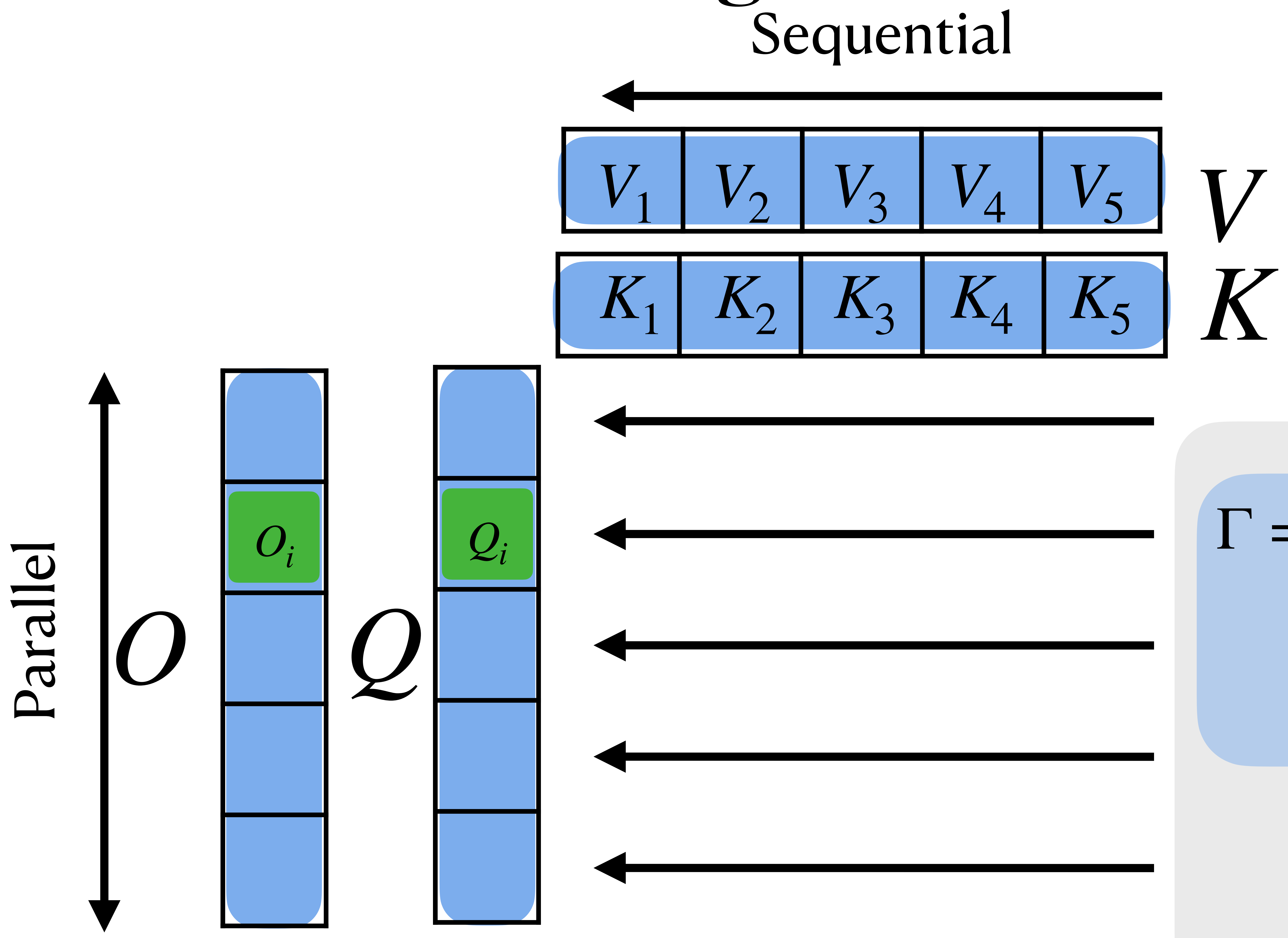
$$Z_i^{(j)} = Z_i^{(j-1)} + \exp(Q_i K_j^\top + D_{ij}) 1$$

Potential Issues of Global Cumsum Implementation

- Trick for computing $d_{ij} = \sum_{l=j+1}^{i} \log f_l$
  - First compute $c_i = \sum_{l=1}^{i} \log f_l$.
  - Then we have $d_{ij} = c_i - c_j$

- Potential issue: **cancellation error** (e.g., $(a + b + c) - (a + b) \neq c$) when computing
  - d_{ij}
  - $\frac{\partial L}{\partial r_t}$, where L is loss and $r_t = \log f_t$
- **Unfortunately it doesn't seem possible to avoid both in the backward pass** (without quadratic memory cost).
- Fine with FP32 and context length 32k. Might be problematic for super long context though (e.g., 10M tokens).

Avoiding Cancellation Error for d_{ij}



$$Y_i^{(j)} \leftarrow Y_i^{(j+1)} + \exp(Q_i K_j^\top) V_j$$

$$Z_i^{(j)} = Z_i^{(j+1)} + \exp(Q_i K_j^\top) 1$$

$$\Gamma = \mathrm{cumsum}(\mathrm{mask}(1(\log f_i)^\top)) \in \mathbb{R}^{B\times B}$$

$$\gamma_i^{(j)} = \gamma_i^{(j+1)} + \Gamma_{1:B,B} \in \mathbb{R}^{B}$$

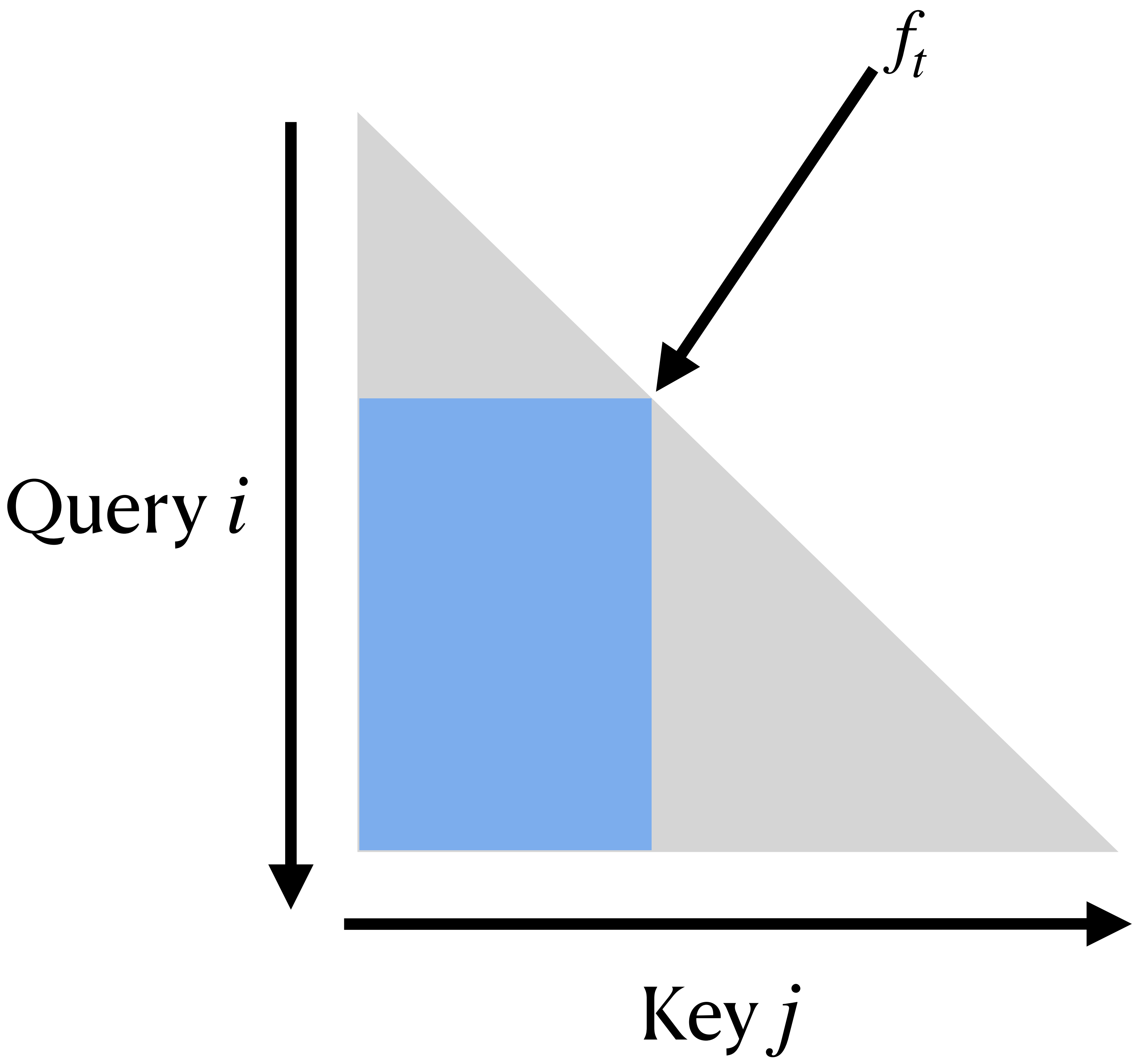
$$D_{ij} = \gamma_i^{(j)} 1^\top - \Gamma \in \mathbb{R}^{B\times B}$$

$$Y_i^{(j)} \leftarrow Y_i^{(j+1)} + \exp(Q_i K_j^\top + D_{ij}) V_j$$

$$Z_i^{(j)} = Z_i^{(j+1)} + \exp(Q_i K_j^\top + D_{ij}) 1$$

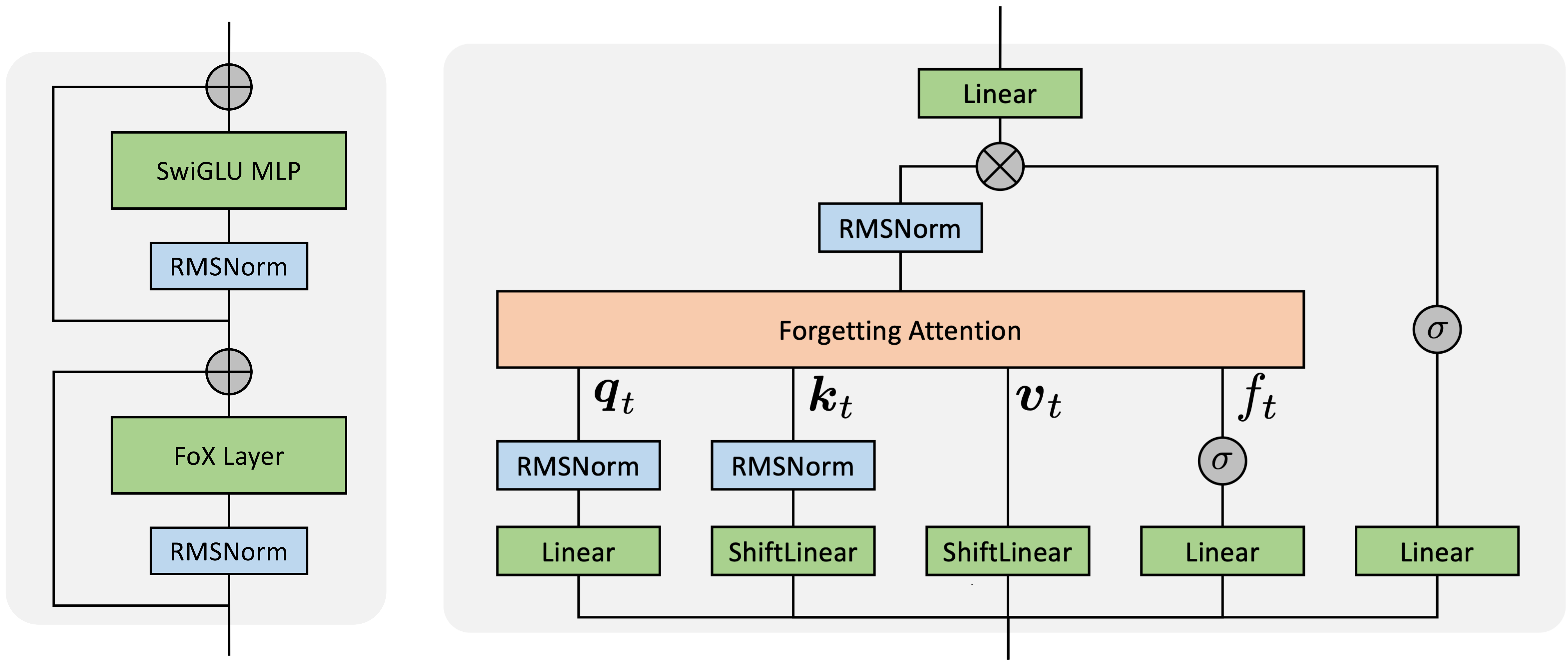
Avoiding Cancellation Error for Gradients of $\log f_t$

- Let $s_{ij} = q_i^\top k_j + d_{ij}$
- Let $r_t = \log f_t$
- Let L be the loss. Then:
  - $\dfrac{\partial L}{\partial r_t} = \sum_{i=t}^{L} \sum_{j=1}^{t-1} \dfrac{\partial L}{\partial s_{ij}}$
- **Requires scanning from left to right** (and thus cancellation error in d_{ij} computation is inevitable)
- **Requires atomic add**



Architecture Design

- **No RoPE** for FoX by default
- **FoX (LLaMA)**: replaces RoPE in the LLaMA arch with forget gates
- **FoX (Pro)**: FoX (LLaMA) plus some components commonly used in recurrent models
    - QK-norm
    - Output normalization
    - Output gate
    - Data-dependent token shift for keys/values (KV-shift)



$$\tilde{\boldsymbol{k}}_t = \boldsymbol{W}_k \boldsymbol{x}_t \in \mathbb{R}^d, \quad \alpha_t^{\text{key}} = \sigma(\boldsymbol{w}_k^\top \boldsymbol{x}_t) \in \mathbb{R}$$

$$\boldsymbol{k}_t = \mathrm{RMSNorm}(\alpha_t^{\text{key}} \tilde{\boldsymbol{k}}_{t-1} + (1 - \alpha_t^{\text{key}}) \tilde{\boldsymbol{k}}_t)$$

Experiments

Core Question

- Our experimental design focuses on one question:
  - **Does FoX forget things in long-context modeling?**
- **No.** What happens in practice is:
  - Some heads do forget quickly (**local heads**)
  - Some heads have extremely slow forgetting (**global heads**)
- Overall:
  - Similar to the Transformer, FoX is great at modeling long sequences
  - In fact, **the longer the sequence, the greater the advantage of FoX over the (RoPE-based) Transformer.**

Overview

- Long-context language modeling
  - **IMPORTANT:** per-token loss analysis
- Needle-in-a-haystack
- Short-context and long-context downstream tasks
- Ablation studies and analyses
  - Different model sizes and training context lengths
  - Components in the Pro architecture
  - RoPE
  - Data-independent/fixed forget gates

Setting

- **Baselines**
  - FoX (Pro)
  - FoX (LLaMA)
  - Transformer (Pro)
  - Transformer (LLaMA)
  - Mamba-2
  - DeltaNet
  - HGRN2

- **Dataset**:
  - LongCrawl64: a pre-tokenized long-sequence subset of RedPajama-v2
- **Model size**:
  - 760M (non-embedding) parameters
- **Tokens:**
  - Training: 48B tokens
  - Eval: 2B tokens

- **Context lengths:**
  - Training: 16K tokens
  - Eval: Up to 64K tokens
- **HParam search:**
  - Search LR in $\{1 \times 10^i, 2 \times 10^i, 5 \times 10^i\}$ for different i's
  - Search head dim in $\{64, 128\}$ for FoX and Transformer

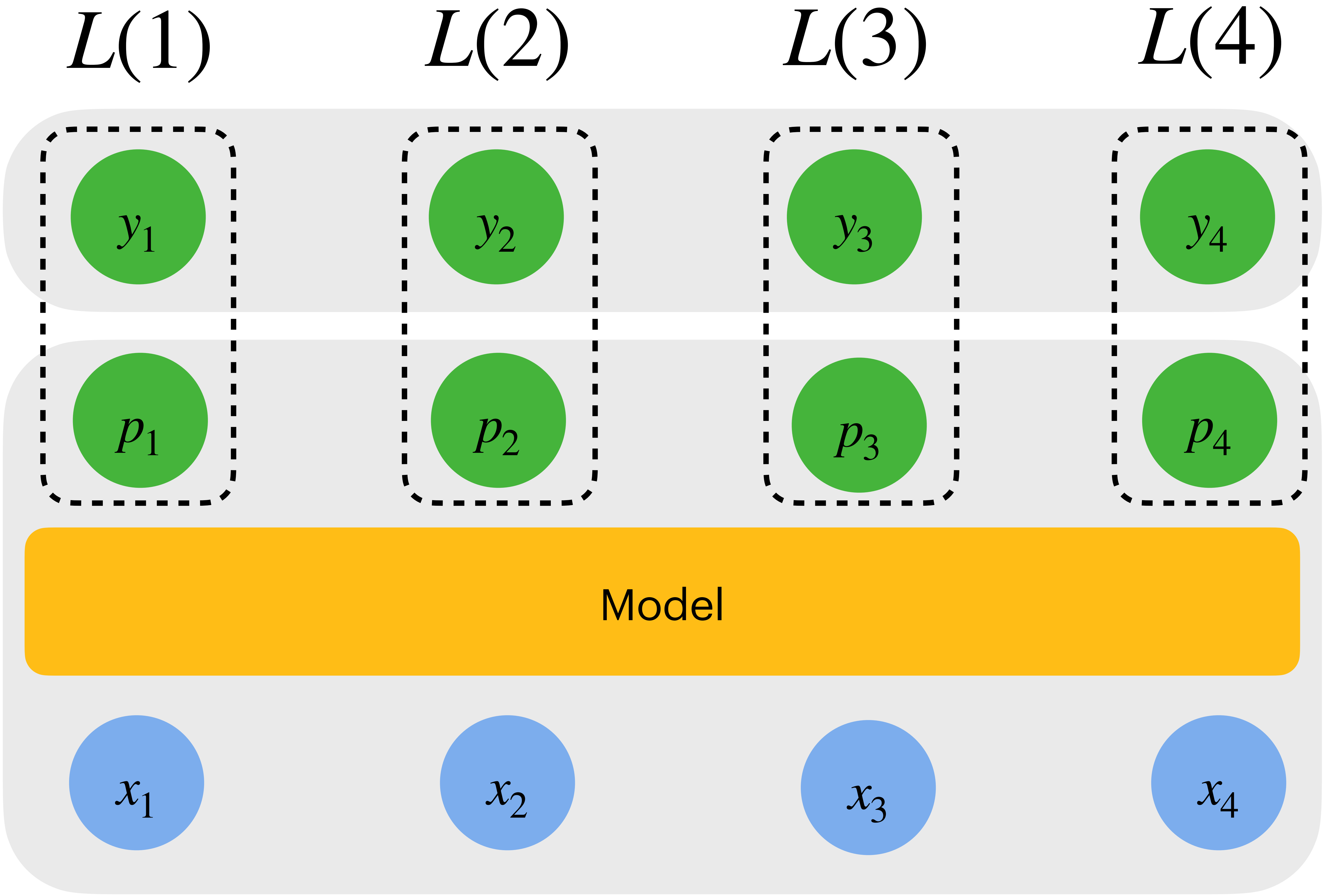
Comments on Optimal Hyperparameters

- FoX prefers **more heads** (and thus smaller head dims) than the Transformer
  - For 760m-param models, $d_{\text{head}} = 64$ for FoX and $d_{\text{head}} = 128$ for the Transformer
  - Shouldn't matter for larger scales because larger models typically have more heads instead of larger head dims
- FoX prefers **higher learning rates** than the Transformer
- Pro architecture models prefer **higher learning rates** than LLaMA architecture models

Model	Learning rate
FoX (Pro)	2×10^{-3}
Transformer (Pro)	1×10^{-3}
FoX (LLaMA)	1×10^{-3}
Transformer (LLaMA)	5×10^{-4}
Mamba-2	2×10^{-3}
HGRN2	2×10^{-3}
DeltaNet	1×10^{-3}

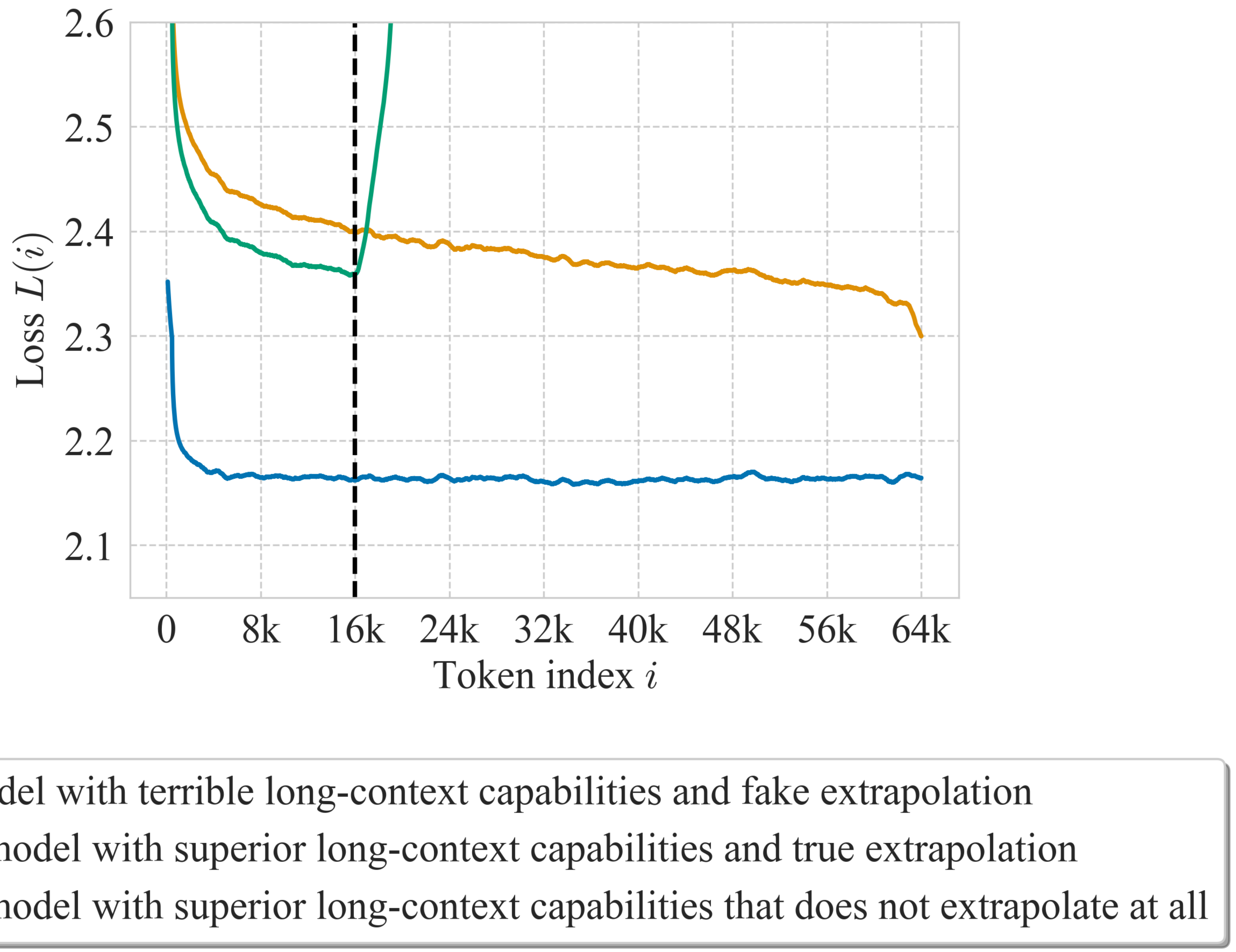
Main Metric: Per-Token Loss

- $L(i) = \frac{1}{N} \sum_{j=1}^{N} -[\log(p_i^{(j)})^\top y_i^{(j)}]$
  - i: the i-th token position
  - j: the j-th sequence
  - $p_i^{(j)} \in \mathbb{R}^{|V|}$: probability over vocab
  - $y_i^{(j)} \in \{0,1\}^{|V|}$: one-hot encoding of language modeling target



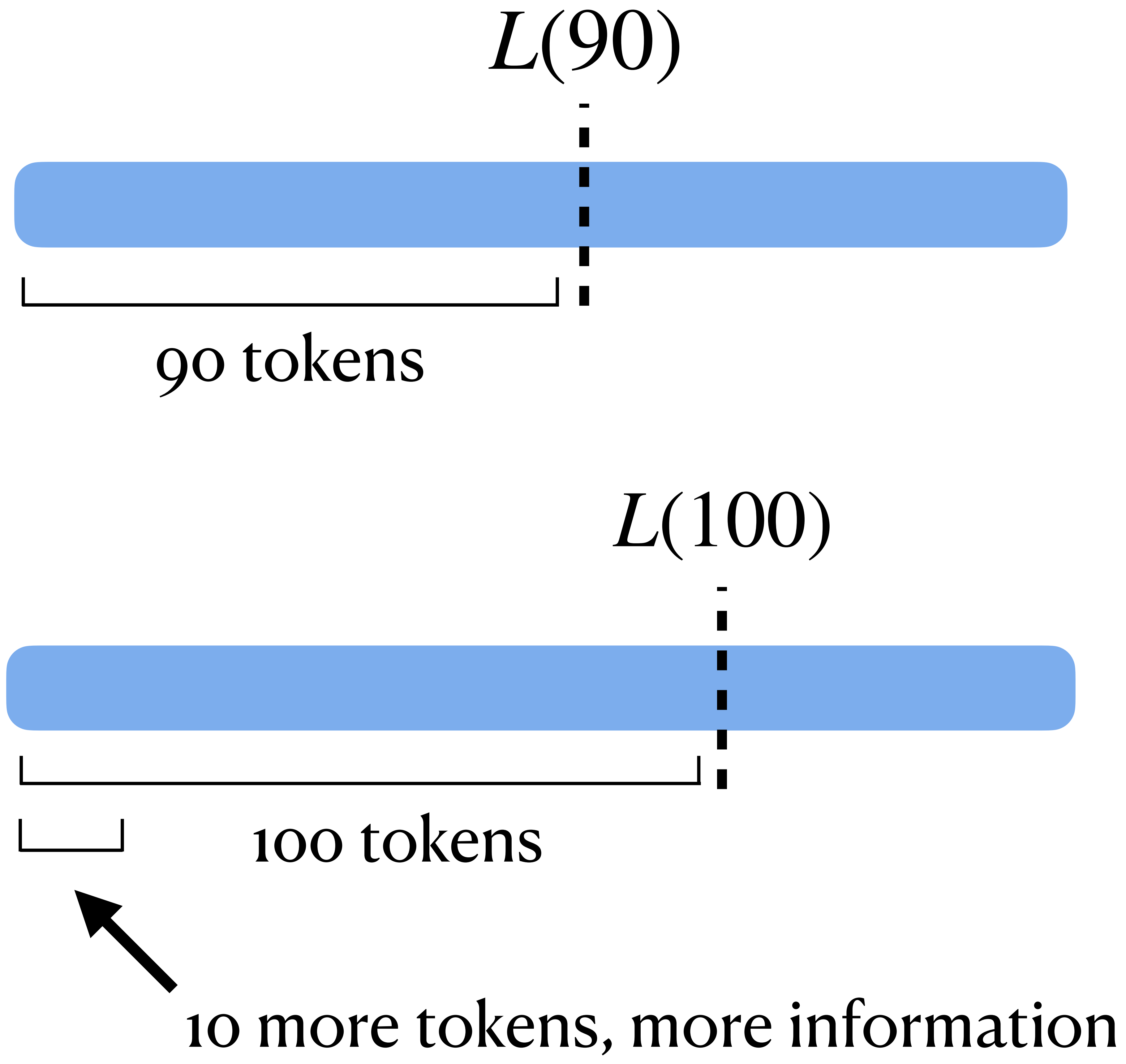
Per-Token Loss

- Per-token loss examples



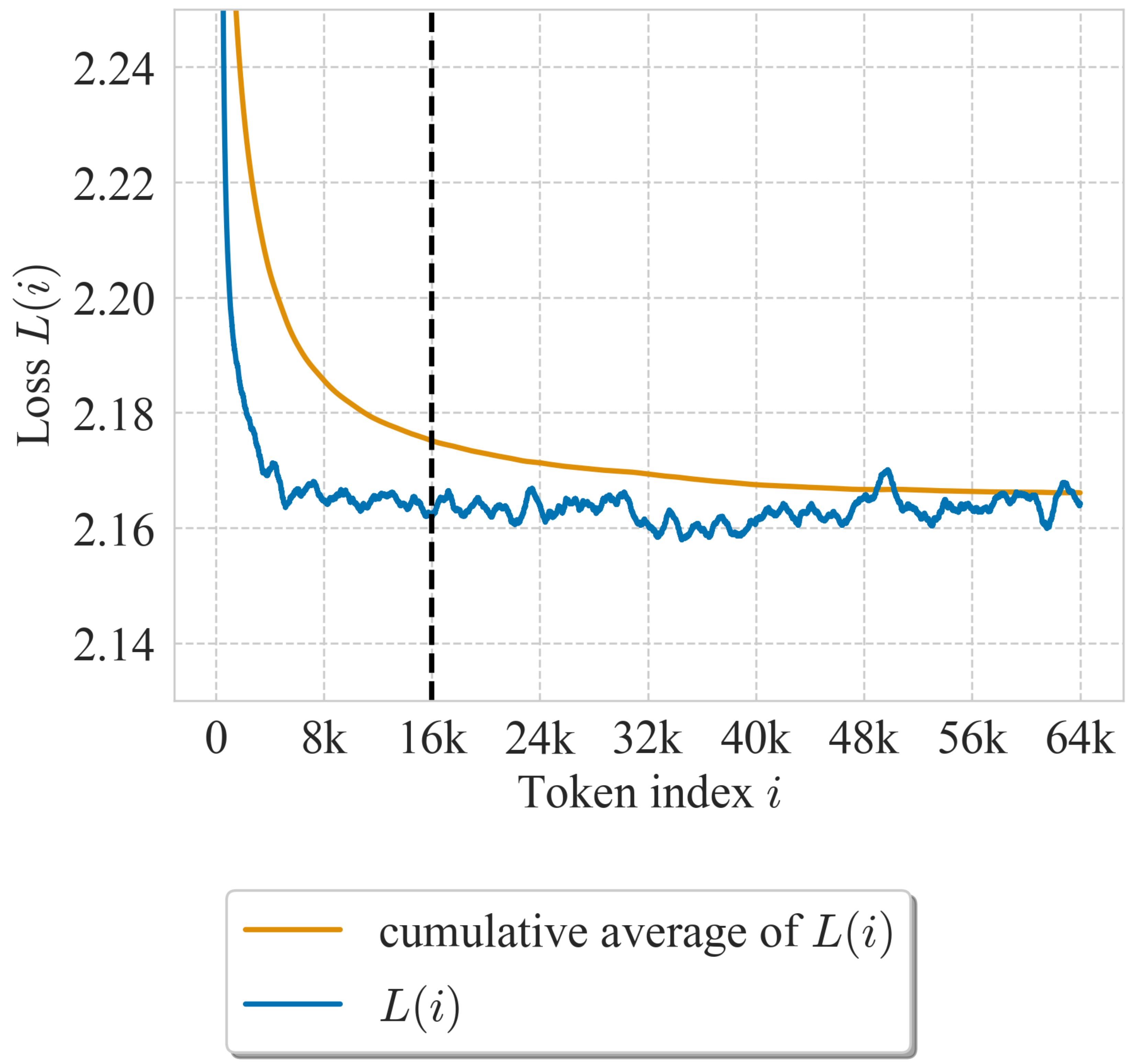
Per-Token Loss and Long Context Modeling

- For a perfect model, $L(i)$ should be monotonically decreasing w.r.t. i
- Given there is no positional bias in the data
  - The **slope** of $L(i)$ at $i \leftrightarrow$ model's ability to utilizes tokens that are i steps earlier
  - $L(i)$ plateaus after index k: the model cannot effective utilize a context with more than k token



Difference Between Per-Token Loss and Perplexity

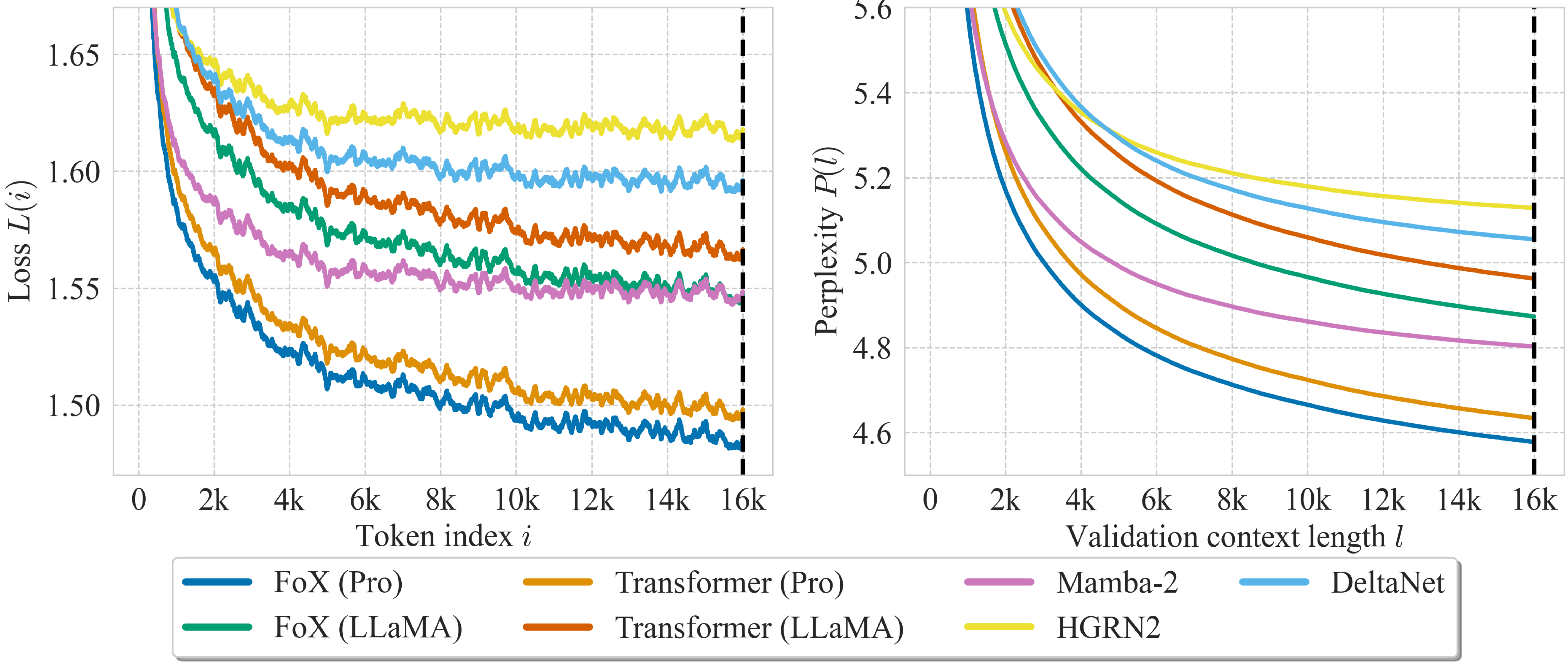
- Per-token loss at token index i:
  - $L(i)$
- Cumulative average of per-token loss over context length l:
  - $L_{\text{cum-avg}}(l) = \sum_{i=1}^{l} L(l)$
- Perplexity over context length l:
  - $P(l) = \exp(\sum_{i=1}^{l} L(i))$



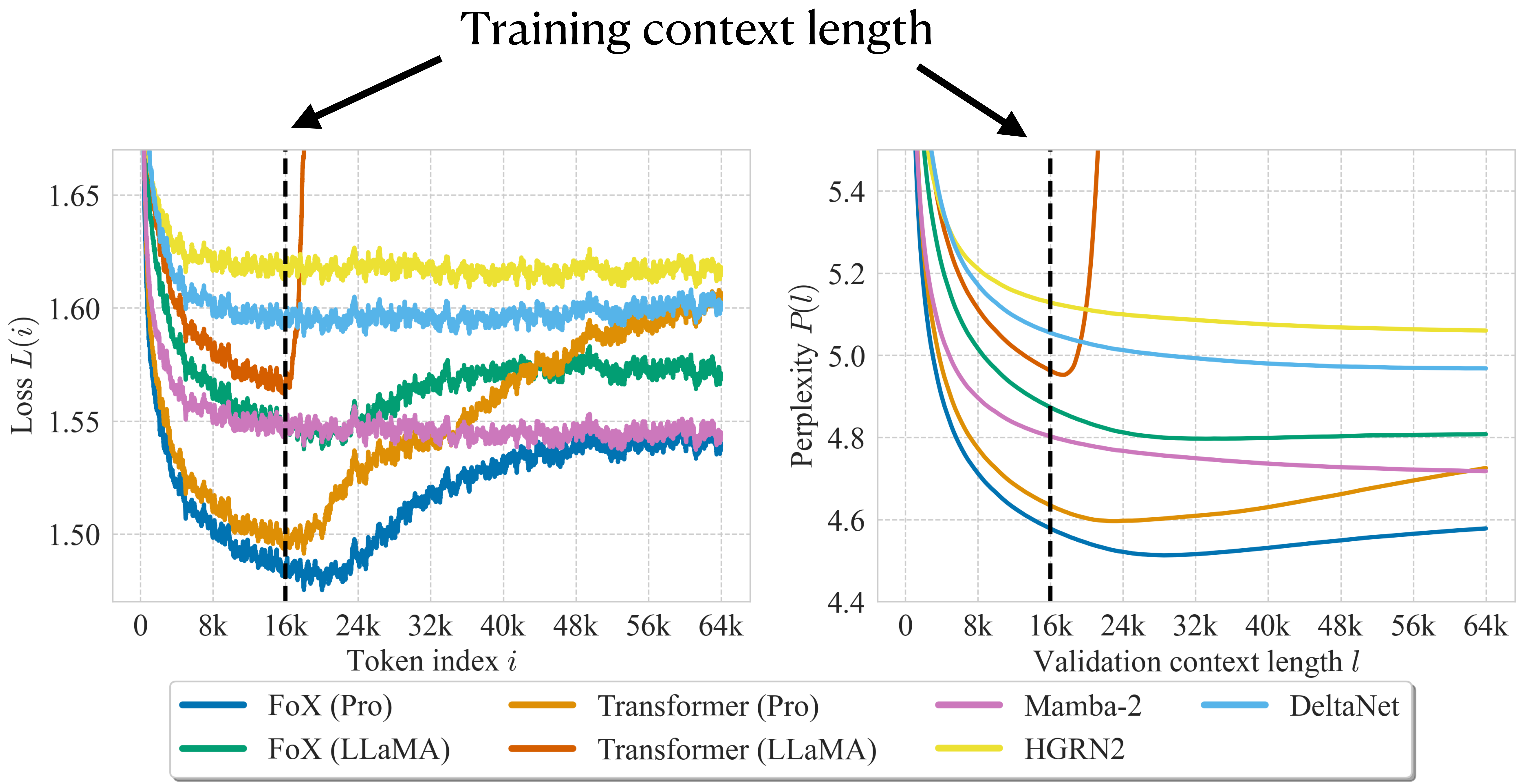
The slope of $P(l)$ could be misleading! Report $L(i)$ instead!

Per-Token Loss

- FoX is better than the (RoPE-based) Transformer
- Similar to the Transformer, FoX has a **monotonically decreasing** per-token loss curve
- Recurrent sequence models cannot fully utilize the long context for their predction

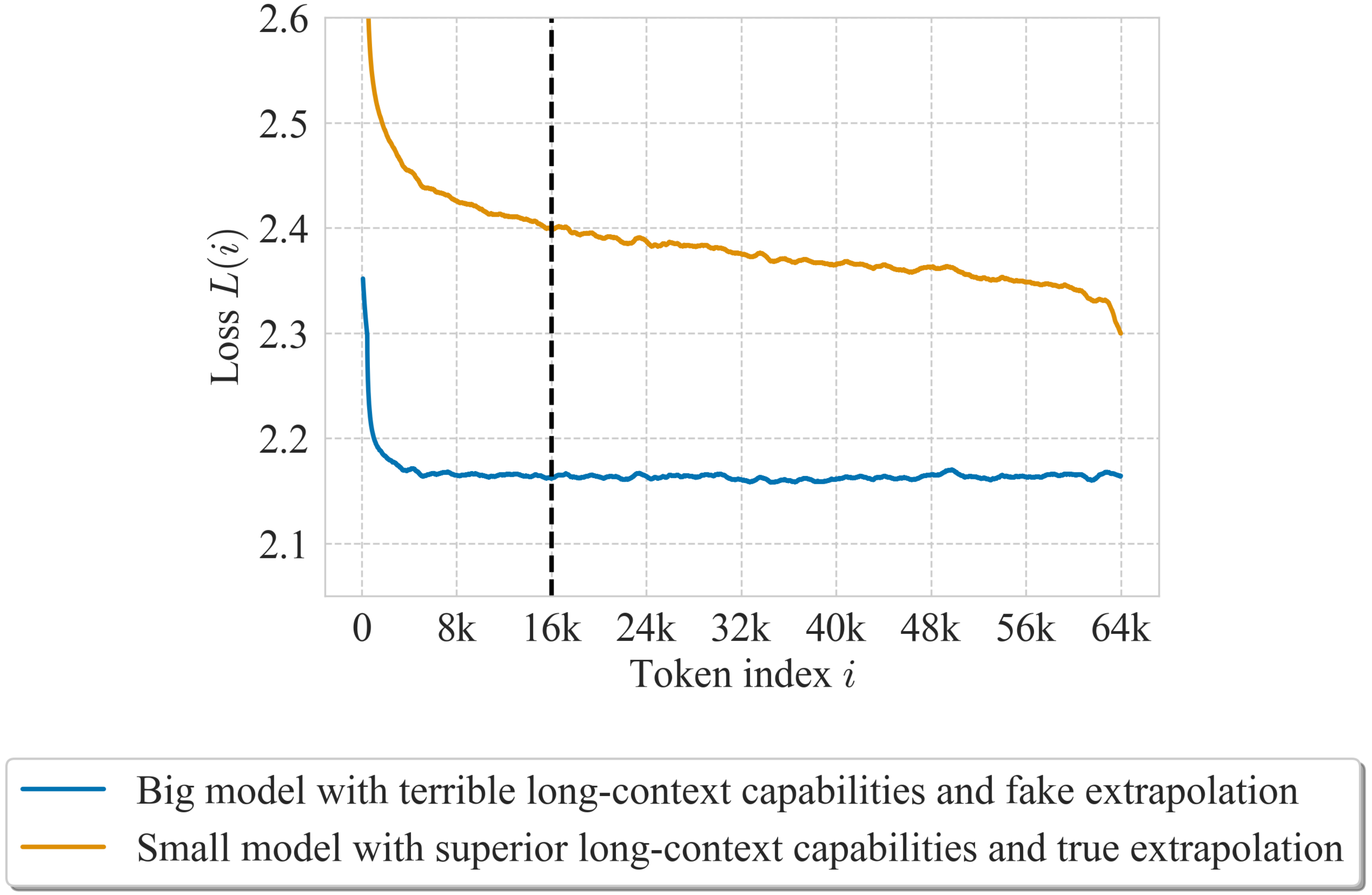


Extrapolation



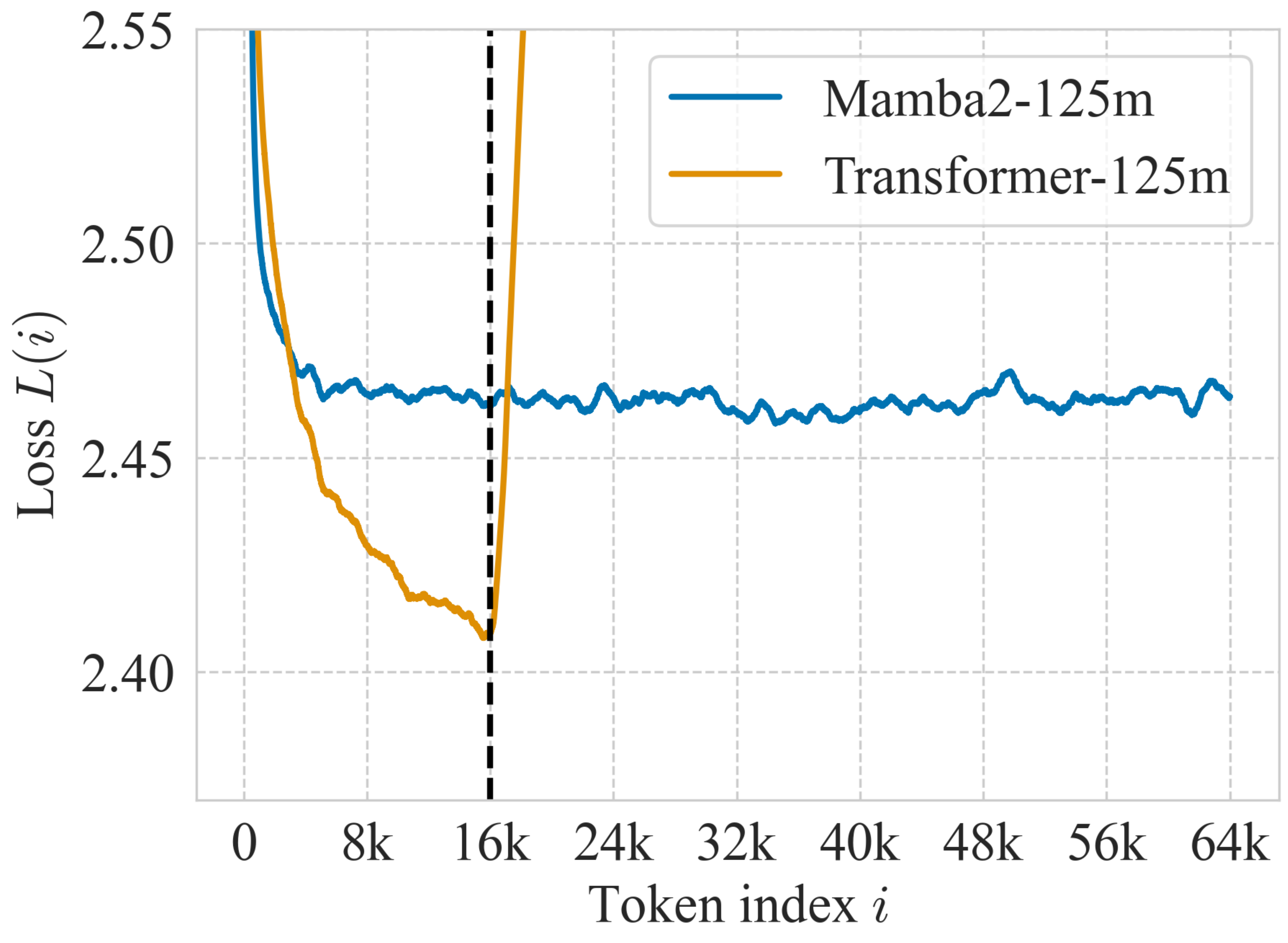
Comments

- I recommend everyone plot $L(i)$, especially for **linear complexity models** (including SWA models).
- For long-context capabilities, what matters is the **slope** $\frac{dL}{di}$, not the (absolute) value $L(i)$
  - $L(10000)$ says nothing about the model's ability to model dependencies that are 10000 tokens long; $\frac{dL}{di}(10000)$ does.
  - One of the reasons loss $\neq$ downstream task performance
- Also matters for extrapolation: **plateauing loss is not "true" extrapolation** (caveat: don't use perplexity $P(l)$)
  - True extrapolation: the extra context should improve prediction;
  - A single-layer SWA models with window size 50 and a training context length of 100 will have perfect "extrapolation"



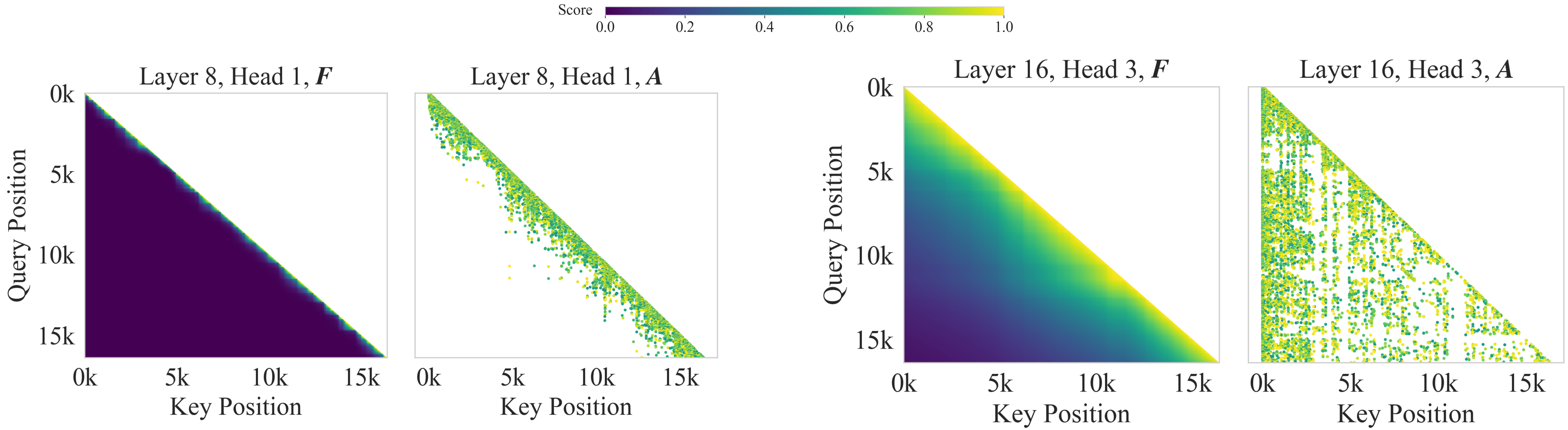
Comments

- You can see a **qualitative** difference between the Transformer and linear complexity models even with small models (e.g., 125m)
  - Actually, most obvious with **small models** with a **long training context length**
- For evaluation: you will need **real long sequences to see this**
  - Concatenating short sequences won't work (see https://manifestai.com/articles/compute-optimal-context-size/)
  - Not sure if this is needed during training (maybe mixing long and short sequences are fine).



Forget Gate Matrix and Attention Map

- Forget gate matrix F: $F_{ij} = \prod_{l=j+1}^{i} f_l$
- Attention matrix: $A = \mathrm{softmax}(QK^\top + \log F)$ (only showing entries larger than 0.5)



Needle-in-a-Haystack

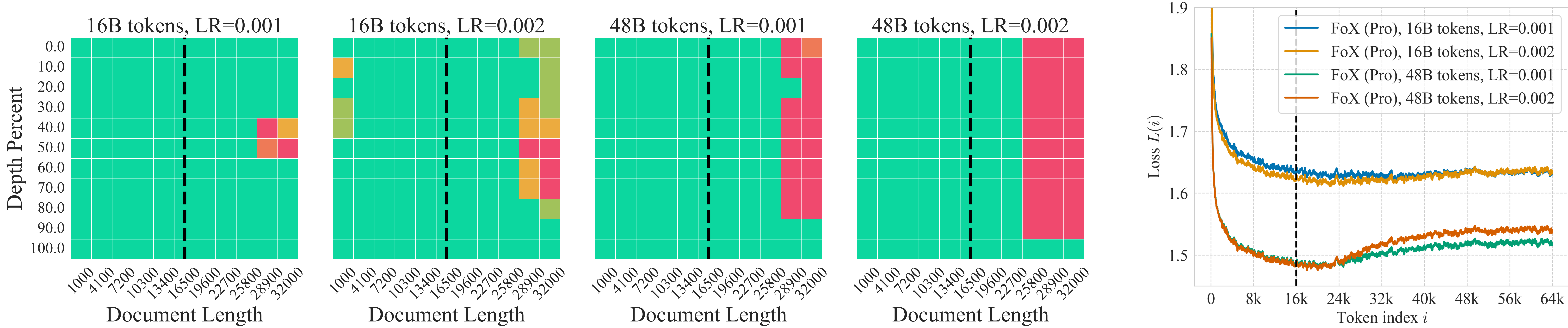
- Standard mode:
  - Needle:
    - `The best thing to do in San Francisco is eat a sandwich and sit in Dolores Park on a sunny day.`
  - Query:
    - `What is the best thing to do in San Francisco? Answer: The best thing to do in San Francisco is`
- Easy mode (Qin et al., 2024):
  - Needle:
    - `What is the best thing to do in San Francisco? Answer: The best thing to do in San Francisco is eat a sandwich and sit in Dolores Park on a sunny day.`
  - Query:
    - `What is the best thing to do in San Francisco? Answer: The best thing to do in San Francisco is`

Needle-in-a-Haystack

Further evidence that FoX can learn not to forget

Extrapolation Behavior is Hyperparameter-Dependent

- In general:
  - Longer training leads to worse extrapolation.
  - Smaller model + longer training context length = better extrapolation; vice versa
  - Not sure about LR
- Extrapolation is nice, but unreliable. **Probably still best to do long-context finetuning**



Short-Context Downstream Tasks

- From Language Model Evaluation Harness

Table 1: Evaluation results on LM-eval-harness. All models have roughly 760M non-embedding parameters and are trained on roughly 48B tokens on LongCrawl64. "acc-n" means length-normalized accuracy. Bold and underlined numbers indicate the best and the second best results, respectively.

Model	**Wiki.** ppl↓	**LMB.** ppl↓	**LMB.** acc↑	**PIQA** acc↑	**Hella.** acc-n↑	**Wino.** acc↑	**ARC-e** acc↑	**ARC-c** acc-n↑	**COPA** acc↑	**OBQA** acc-n↑	**SciQA** acc↑	**BoolQ** acc↑	**Avg** ↑
FoX (Pro)	**23.04**	**14.91**	**42.75**	**64.09**	**38.39**	**52.33**	**52.23**	**26.54**	**71.00**	29.80	**85.10**	46.57	**50.88**
Transformer (Pro)	<u>24.12</u>	<u>16.16</u>	<u>41.47</u>	<u>64.04</u>	<u>36.60</u>	49.72	<u>51.98</u>	25.26	62.00	29.20	<u>82.80</u>	**60.86**	<u>50.39</u>
FoX (LLaMA)	26.45	18.27	40.17	63.44	35.17	<u>51.78</u>	49.66	25.09	69.00	28.00	81.90	54.04	49.82
Transformer (LLaMA)	28.14	22.34	38.27	63.22	34.20	49.49	47.98	24.49	66.00	29.40	78.90	<u>58.93</u>	49.09
Mamba-2	28.20	21.05	36.50	63.17	35.86	50.59	49.96	<u>25.60</u>	**71.00**	**31.00**	80.90	57.49	50.21
HGRN2	30.57	20.14	38.60	63.49	34.94	<u>51.78</u>	50.13	25.51	66.00	<u>30.00</u>	75.60	58.41	49.45
DeltaNet	29.17	29.14	34.27	62.73	33.28	50.28	47.39	24.32	<u>70.00</u>	29.00	74.30	54.37	47.99

Long-Context Downstream Tasks

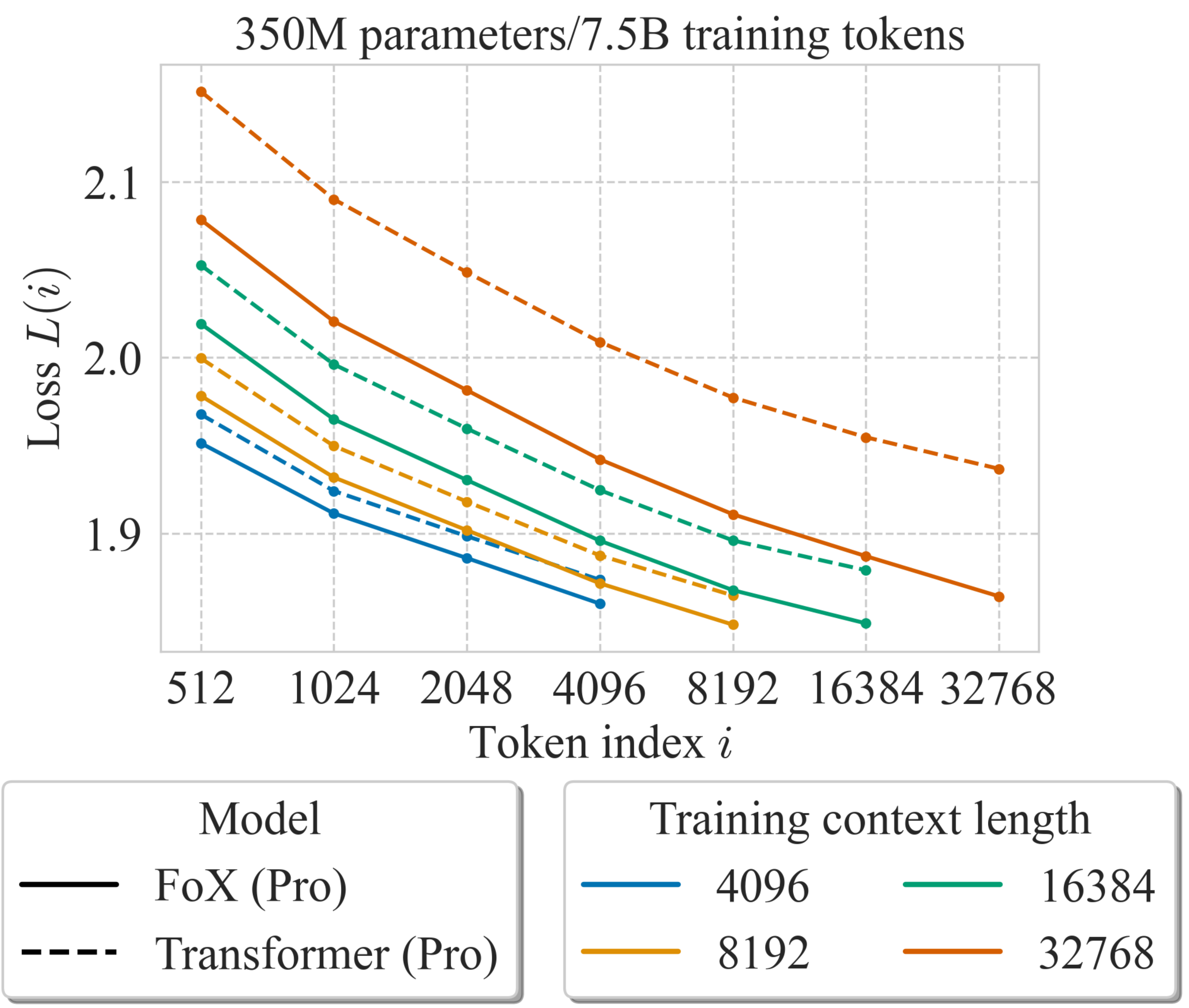
- From LongBench-v1

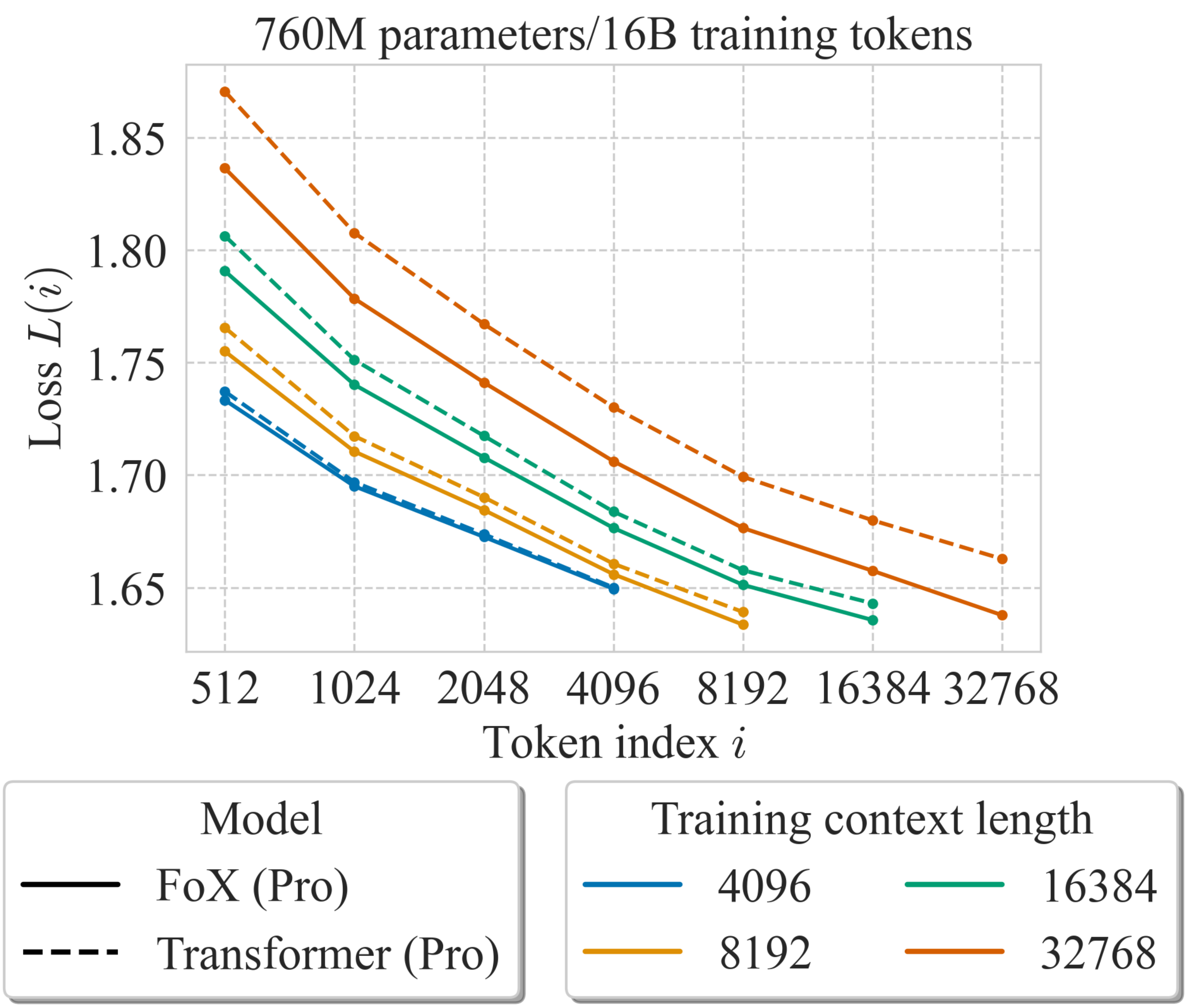
Table 2: Evalution results on LongBench. All models have roughly 760M non-embedding parameters and are trained on roughly 48B tokens on LongCrawl64. Bold and underlined numbers indicate the best and the second-best results, respectively.

Model	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Code	
	NarrativeQA	Qasper	MFQA	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SamSum	LCC	RepoBench-P
FoX (Pro)	**13.38**	<u>18.88</u>	**28.73**	<u>15.27</u>	**25.39**	**6.49**	**22.71**	**13.51**	**12.27**	**63.5**	<u>37.36</u>	<u>22.74</u>	10.9	9.1
Transformer (Pro)	<u>11.42</u>	**21.54**	22.89	**19.58**	<u>22.65</u>	<u>6.09</u>	<u>21.92</u>	10.7	8.11	55.0	**40.67**	**30.66**	10.79	14.25
FoX (LLaMA)	10.47	14.81	<u>24.71</u>	13.03	21.58	5.25	20.05	10.97	4.86	<u>61.5</u>	34.48	19.13	7.69	8.12
Transformer (LLaMA)	11.11	13.5	21.52	9.42	21.33	4.32	18.53	8.43	<u>10.99</u>	51.5	28.41	19.17	8.21	14.06
Mamba-2	10.65	11.26	16.98	11.59	16.69	5.0	9.31	11.22	10.89	28.5	15.6	16.19	12.07	<u>15.17</u>
HGRN2	8.78	10.94	18.66	7.78	15.29	4.32	6.13	12.19	7.83	16.5	14.46	6.37	**18.17**	**16.62**
DeltaNet	9.36	9.76	16.49	6.57	15.09	2.76	8.19	<u>12.3</u>	7.62	35.5	17.57	18.42	<u>12.24</u>	3.94

Model Size/Training Context Length

- **Note**: these experiments use LRs tuned for Transformer (LLaMA) with context length 16k. Under optimal LRs the gaps will likely be larger
- **Hypothesis**: the benefits of forget gates depend on the **ratio** between model size and training context length
  - Larger models can better model long contexts, thus forgetting is less important





Ablations

- Everything in the Pro architecture is useful

Table 3: Ablation experiments for FoX. We use 360M-parameter models trained on 7.5B tokens on LongCrawl64. The perplexity is measured over a validation context length of 16384 tokens. For the bottom half, all addition (+) or removal (-) of components are relative to FoX (Pro).

Model	**RoPE**	**Forget gate**	**QK-norm**	**Output gate**	**Output norm**	**KV-shift**	**Perplexity**
Transformer (LLaMA) w/o RoPE							29.30
Transformer (LLaMA)	✓						7.49
	✓	✓					7.19
FoX (LLaMA)		✓					7.25
		✓	✓				7.08
		✓	✓	✓			6.88
		✓	✓	✓	✓		6.80
FoX (Pro)		✓	✓	✓	✓	✓	6.62
- QK-norm		✓		✓	✓	✓	6.79
- output gate		✓	✓		✓	✓	6.86
- output norm		✓	✓	✓		✓	6.69
- KV-shift		✓	✓	✓	✓		6.80
+ RoPE	✓	✓	✓	✓	✓	✓	6.63
- forget gate + RoPE (i.e. Transformer (Pro))	✓		✓	✓	✓	✓	6.82
- forget gate			✓	✓	✓	✓	7.40

Poor performance if removing both forget gates and RoPE

Ablations

- Unclear whether RoPE is still useful in FoX; certainly not **necessary**

Table 3: Ablation experiments for FoX. We use 360M-parameter models trained on 7.5B tokens on LongCrawl64. The perplexity is measured over a validation context length of 16384 tokens. For the bottom half, all addition (+) or removal (-) of components are relative to FoX (Pro).

Model	**RoPE**	**Forget gate**	**QK-norm**	**Output gate**	**Output norm**	**KV-shift**	**Perplexity**
Transformer (LLaMA) w/o RoPE							29.30
Transformer (LLaMA)	✓						7.49
	✓	✓					7.19
FoX (LLaMA)		✓					7.25
		✓	✓				7.08
		✓	✓	✓			6.88
		✓	✓	✓	✓		6.80
FoX (Pro)		✓	✓	✓	✓	✓	6.62
- QK-norm		✓		✓	✓	✓	6.79
- output gate		✓	✓		✓	✓	6.86
- output norm		✓	✓	✓		✓	6.69
- KV-shift		✓	✓	✓	✓		6.80
+ RoPE	✓	✓	✓	✓	✓	✓	6.63
- forget gate + RoPE (i.e. Transformer (Pro))	✓		✓	✓	✓	✓	6.82
- forget gate			✓	✓	✓	✓	7.40

FoX (LLaMA) + RoPE

FoX (LLaMA)

FoX (Pro)

FoX (Pro) + RoPE

Data-Independent Forget Gate

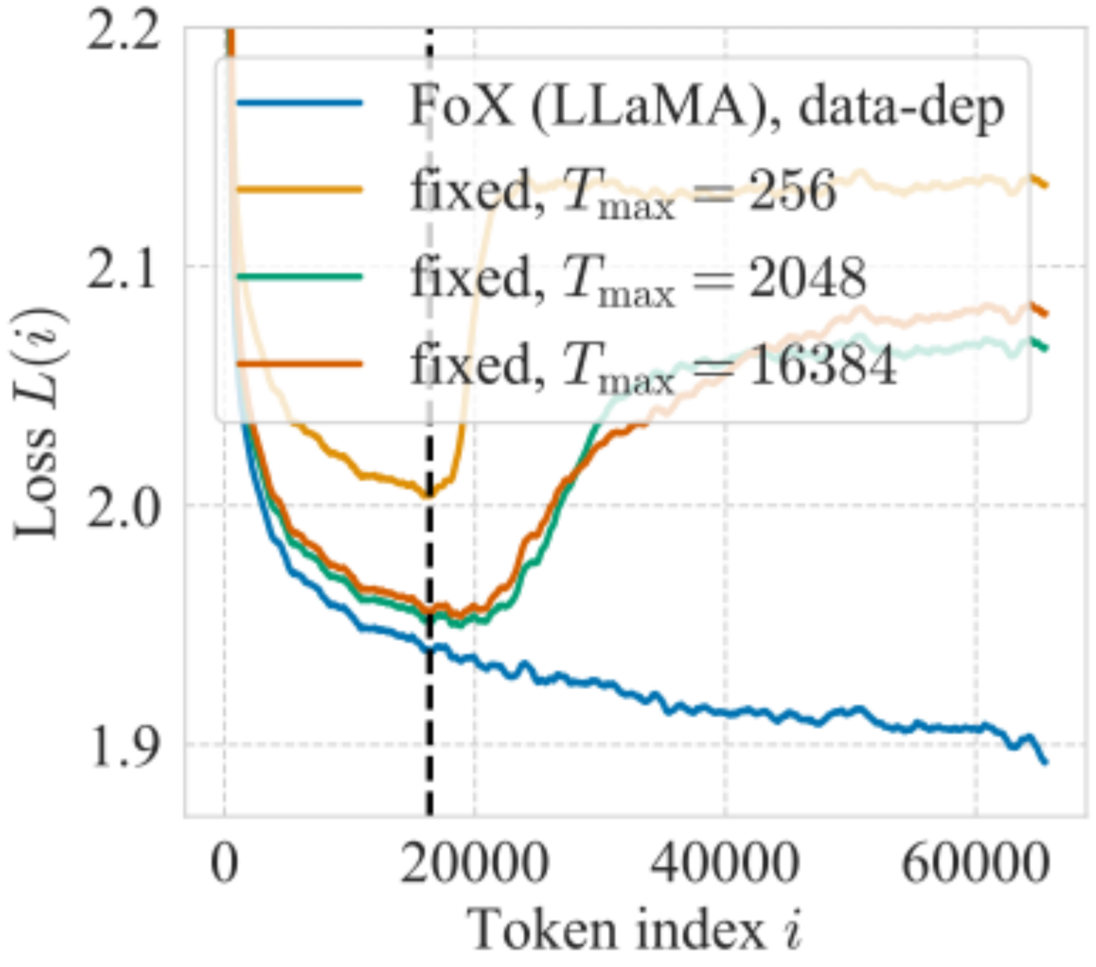
- Data-dependent:
  - $f_t^{(h)} = \sigma(w_f^{(h)} x_t + b_f^{(h)})$
- Data-independent:
  - $f_t^{(h)} = \sigma(b_f^{(h)})$
- Fixed (equivalent to ALiBi):
  - $f_t^{(h)} = \sigma(b_f^{(h)})$, with fixed $b_f^{(h)}$

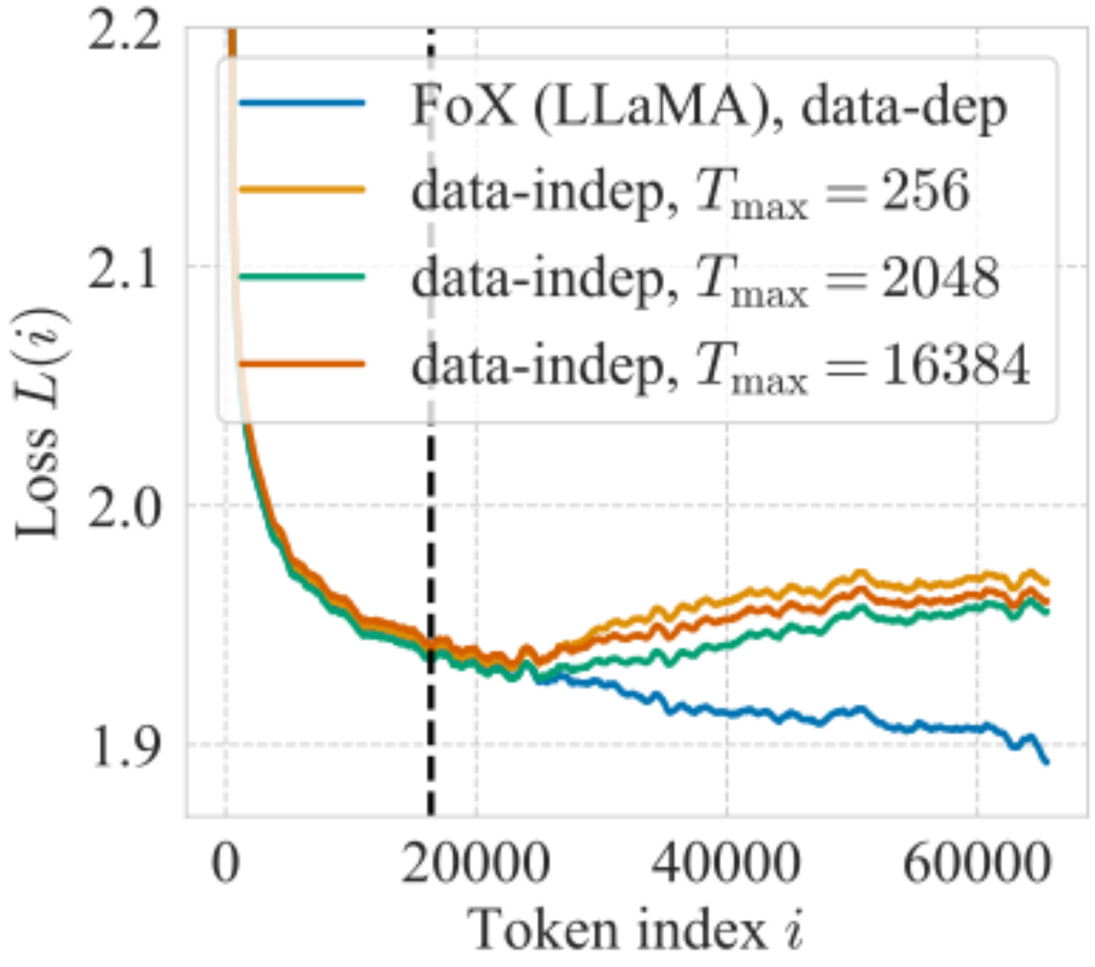
Data-Independent Forget Gate

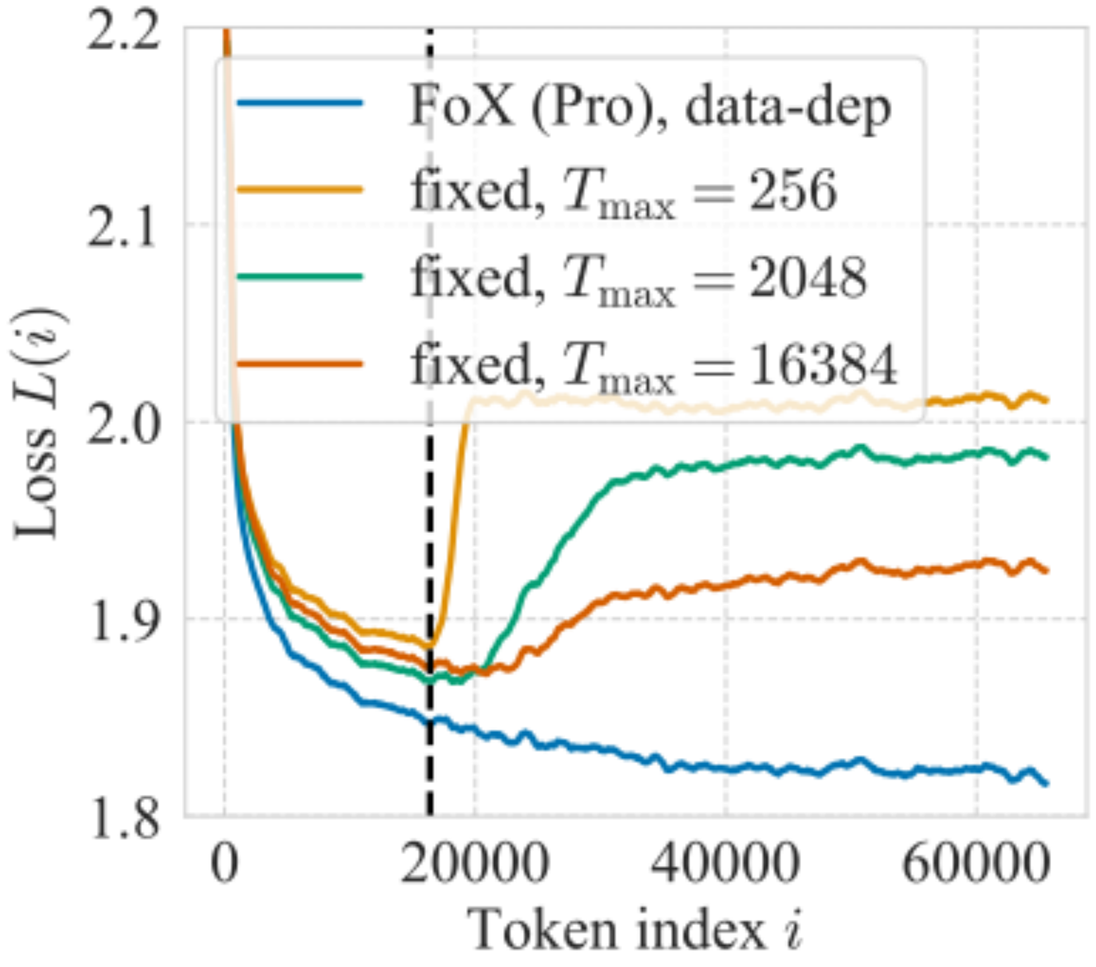
- For data-indep and fixed forget gates, $\{b_f^{(h)}\}_{h=1}^{H}$ are initialized with two hparams $(T_{\min}, T_{\max})$ such that
    - Fixed forget gates initialized with $(T_{\min}, T_{\max})$ is equivalent to ALiBi with maximum slope $\dfrac{1}{T_{\min}}$ and minimum slope $\dfrac{1}{T_{\max}}$

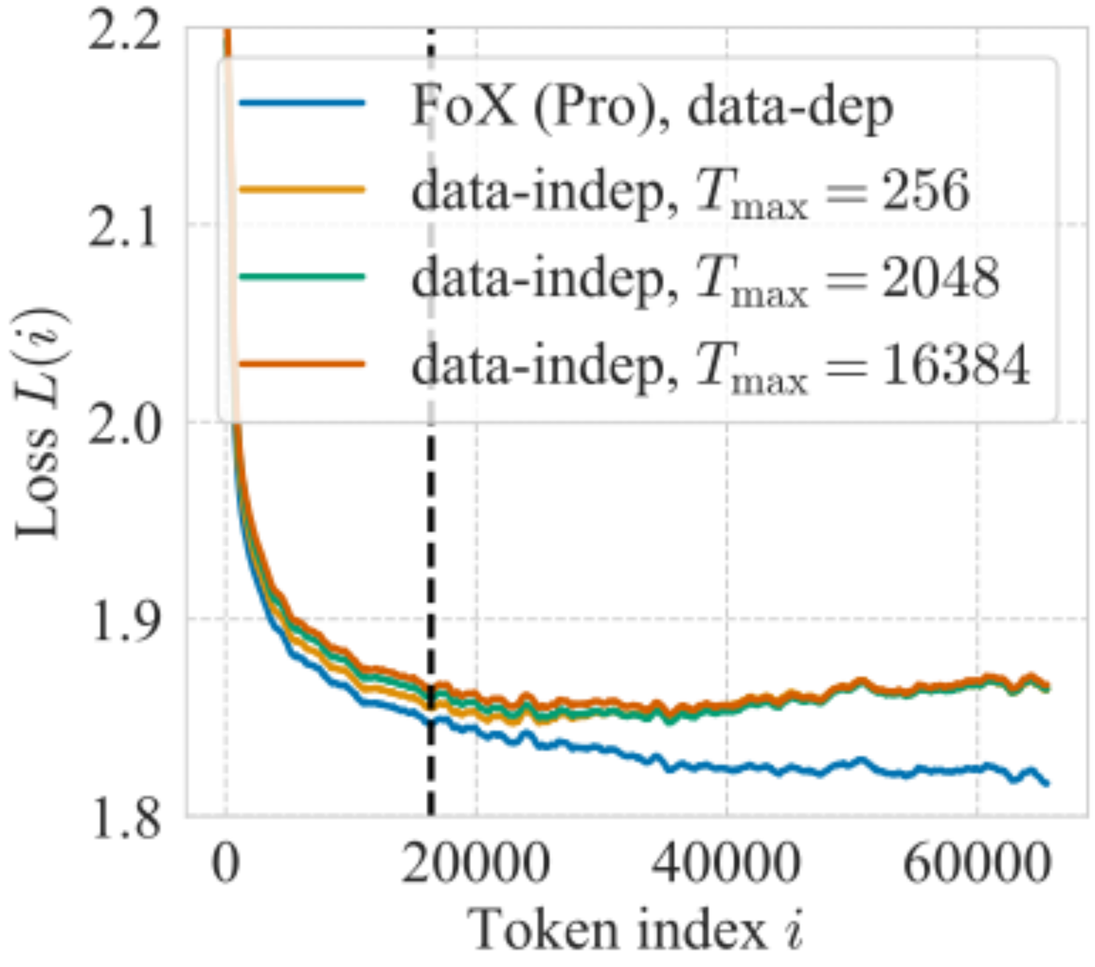
Data-Dependent vs Data-Independent vs Fixed

- We fix $T_{\text{min}} = 2$ and vary T_{max}
- Data-dependent forget gates always the best, and **hyperparameter-free**.









Future Directions

Future directions

- Try FoX at larger scales
- Make this bi-directional
- Long-context fine-tuning: should be great because
  - Forget gates are learnable and data-dependent: should adapt quickly
  - FoX has better length extrapolation: stable and faster learning during fine-tuning
- Adding forget gates to **pretrained models** (e.g., LLaMA3), and the finetune
  - It should adapt quickly
- **Saving computation based on forget gate values (work in progress).**

Many Heads Are Local

- If a head only uses a local context, no need to waste compute on distant tokens (they don't affect the output anyways)

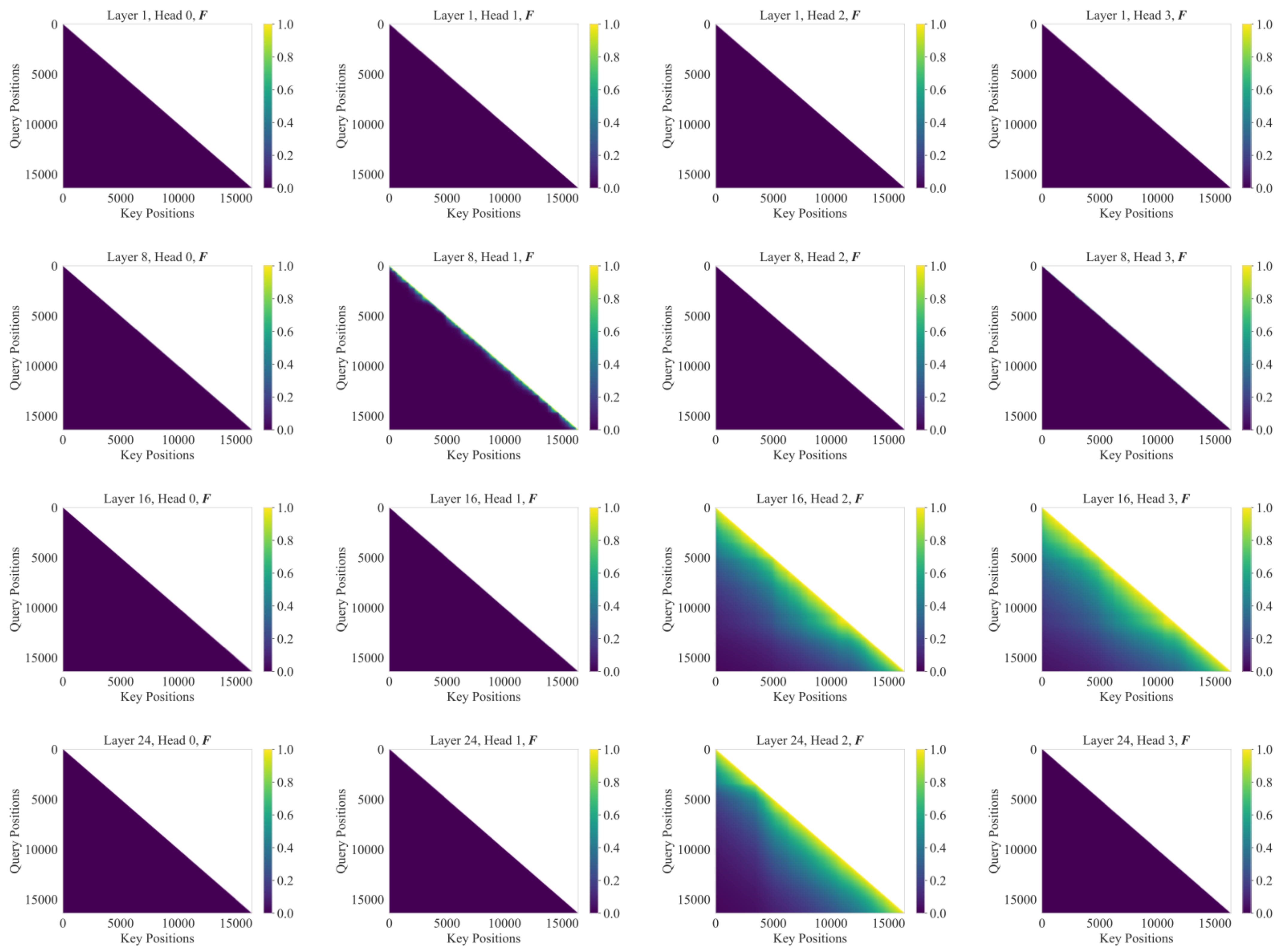


Figure 27: Visualization of the forget gate weight matrix $\boldsymbol{F}$ from 16 heads in 4 different layers. These results use FoX (Pro).

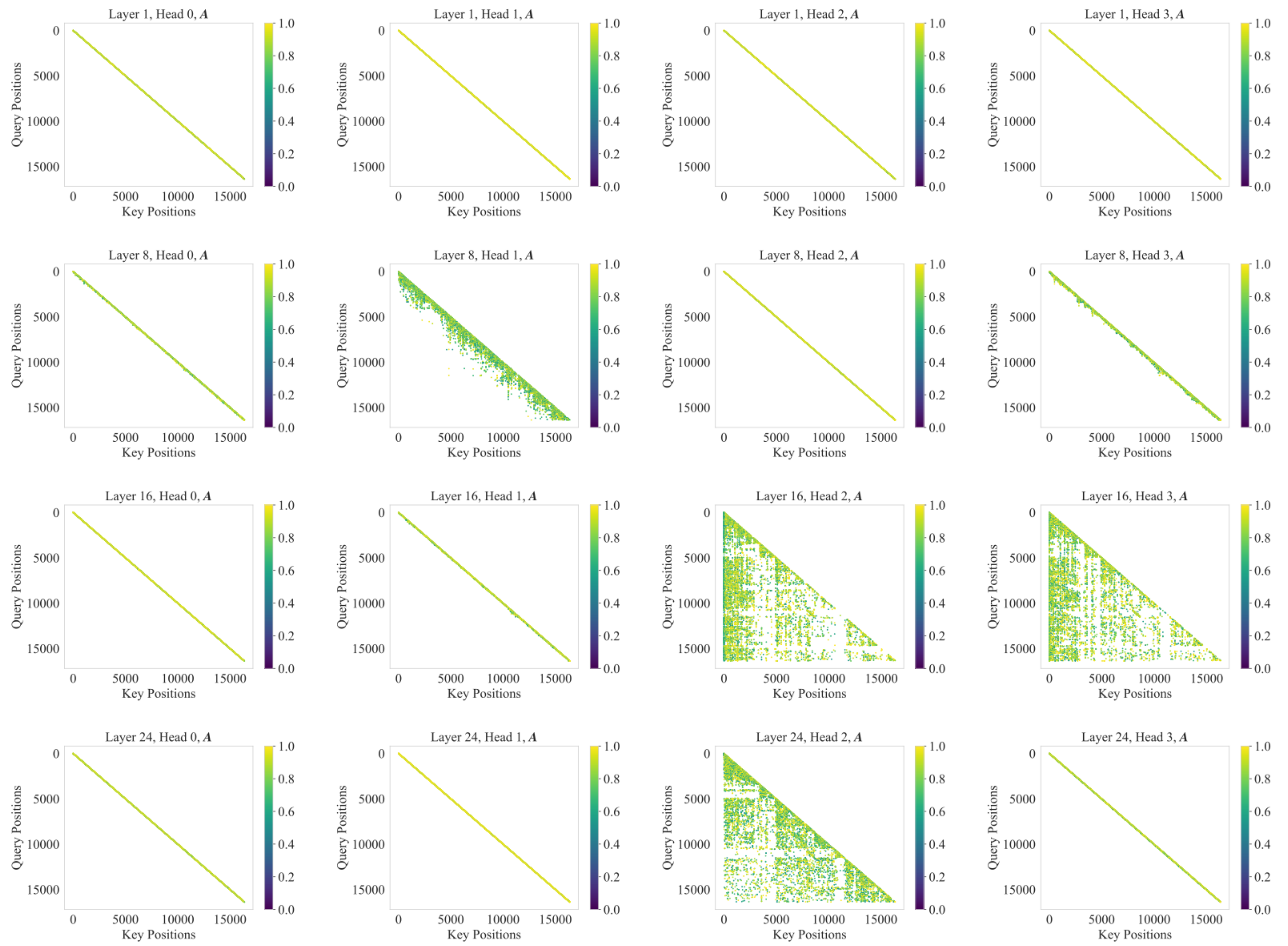


Figure 28: Visualization of the attention score matrix $\boldsymbol{A}$ from 16 heads in 4 different layers. These results use FoX (Pro).

Adaptive Block Skipping

Very cheap to compute

- $d_{ij} = c_i - c_j$, where

$$c_i = \sum_{l=1}^{i} \log f_l$$

$$o_i = \frac{\sum_{j=1}^{i} \exp(q_i^\top k_j + d_{ij}) v_j}{\sum_{j=1}^{i} \exp(q_i^\top k_j + d_{ij})}$$

- $d_{ii} = 0$ by definition. So if d_{ij} is very negative for some $j \neq i$, we can safely ignore $\exp(q_i^\top k_j + d_{ij})v_j$
  - Guaranteed to lossless if $q_i^\top k_j$ is bounded, which is true if we use QK-norm.
- 125M model, 16K training context lengths, **30% throughput improvement**
  - Saved attention FLOPs should be way larger than 30%.

Thanks!

References

- Klaus Greff, Rupesh K Srivastava, Jan Koutn´ık, Bas R Steunebrink, and J¨urgen Schmidhuber. Lstm: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10): 2222–2232, 2016.
- Jos Van Der Westhuizen and Joan Lasenby. The unreasonable effectiveness of the forget gate. arXiv preprint arXiv:1804.04849, 2018.
- Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. arXiv preprint arXiv:2312.06635, 2023.
- Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752, 2023.
- Jacob Buckman, Carles Gelada, and Sean Zhang. Symmetric Power Transformers, 2024.
- Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. arXiv preprint arXiv:2103.02143, 2021.
- Choromanski, Krzysztof, et al. "Rethinking attention with performers." arXiv preprint arXiv:2009.14794 (2020).
- Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. arXiv preprint arXiv:2406.06484, 2024.
- Schlag, Imanol, Kazuki Irie, and Jürgen Schmidhuber. "Linear transformers are secretly fast weight programmers." International Conference on Machine Learning. PMLR, 2021.
- Munkhdalai, Tsendsuren, et al. "Metalearned neural memory." Advances in Neural Information Processing Systems 32 (2019).

References

- S4: Gu, Albert, Karan Goel, and Christopher Ré. "Efficiently modeling long sequences with structured state spaces." arXiv preprint arXiv:2111.00396 (2021).
- S5: Smith, Jimmy TH, Andrew Warrington, and Scott W. Linderman. "Simplified state space layers for sequence modeling." arXiv preprint arXiv:2208.04933 (2022).
- LRU: Orvieto, Antonio, et al. "Resurrecting recurrent neural networks for long sequences." International Conference on Machine Learning. PMLR, 2023.
- HGRN: Qin, Zhen, Songlin Yang, and Yiran Zhong. "Hierarchically gated recurrent neural network for sequence modeling." Advances in Neural Information Processing Systems 36 (2024).
- Mega: Ma, Xuezhe, et al. "Mega: moving average equipped gated attention." arXiv preprint arXiv:2209.10655 (2022).
- RG-LRU: De, Soham, et al. "Griffin: Mixing gated linear recurrences with local attention for efficient language models." arXiv preprint arXiv:2402.19427 (2024).
- Gated RFA: Peng, Hao, et al. "Random feature attention." *arXiv preprint arXiv:2103.02143* (2021).
- RWKV-4: Peng, Bo, et al. "Rwkv: Reinventing rnns for the transformer era." arXiv preprint arXiv:2305.13048 (2023).
- linear attention: Katharopoulos, Angelos, et al. "Transformers are rnns: Fast autoregressive transformers with linear attention." International conference on machine learning. PMLR, 2020.
- GLA: Yang, Songlin, et al. "Gated linear attention transformers with hardware-efficient training." arXiv preprint arXiv:2312.06635 (2023).
- Mamba-2: Dao, Tri, and Albert Gu. "Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality." arXiv preprint arXiv:2405.21060 (2024).
- RetNet: Sun, Yutao, et al. "Retentive network: A successor to transformer for large language models." arXiv preprint arXiv:2307.08621 (2023).
- HGRN2: Qin, Zhen, et al. "Hgrn2: Gated linear rnns with state expansion." arXiv preprint arXiv:2404.07904 (2024).
- RWKV-6: Peng, Bo, et al. "Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence." arXiv preprint arXiv:2404.05892 (2024).
- xLSTM: Beck, Maximilian, et al. "xLSTM: Extended Long Short-Term Memory." arXiv preprint arXiv:2405.04517 (2024).
- ALiBi: Press, Ofir, Noah A. Smith, and Mike Lewis. "Train short, test long: Attention with linear biases enables input length extrapolation." arXiv preprint arXiv:2108.12409 (2021).
- RoPE: Su, Jianlin, et al. "Roformer: Enhanced transformer with rotary position embedding." *Neurocomputing* 568 (2024): 127063.
- Gated DeltaNet: Yang, Songlin, Jan Kautz, and Ali Hatamizadeh. "Gated Delta Networks: Improving Mamba2 with Delta Rule." *arXiv preprint arXiv:2412.06464* (2024).
- SBA: Tan, Shawn, et al. "Stick-breaking Attention." *arXiv preprint arXiv:2410.17980* (2024).