Scaling Context Requires Rethinking Attention

Jacob Buckman
Carles Gelada
Sean Zhang

July 2025

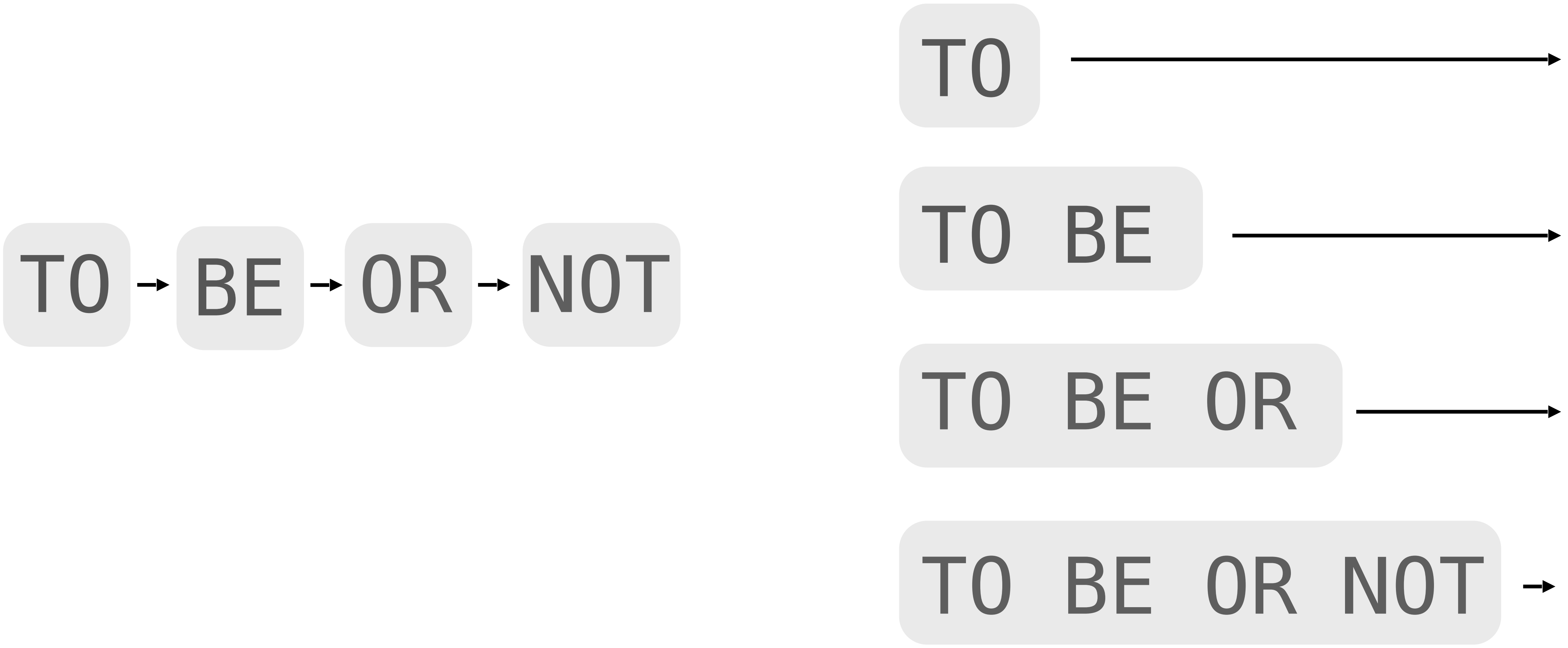
Why did transformers **win**?

- GPU-friendly
- State gets large

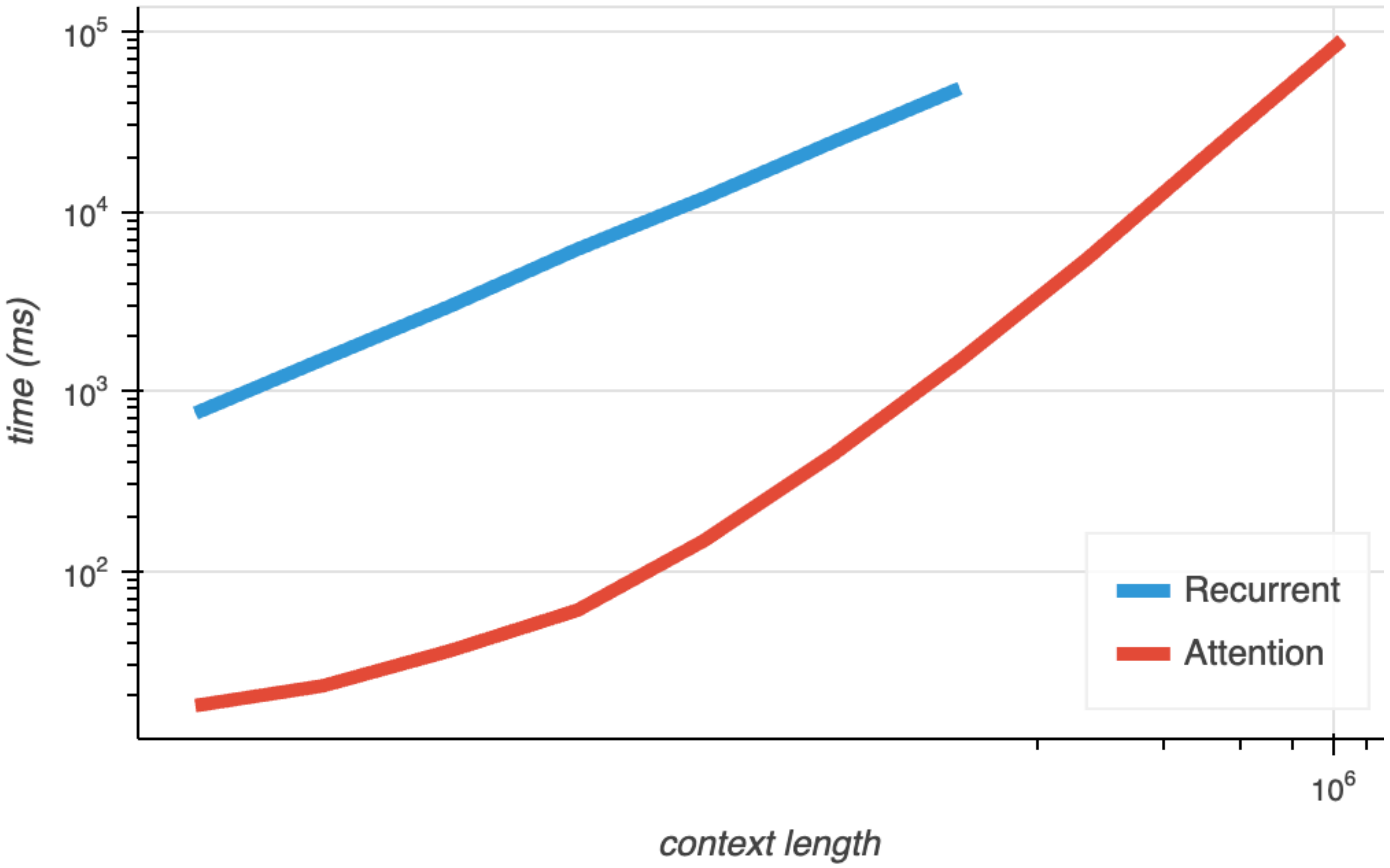
Why did transformers **win**?

- GPU-friendly
- State gets large

	Classic RNN (e.g. LSTM, GRU)	**Attention** (e.g. Transformer)
FLOPs	t	$t^2/2$
Big matmuls?	✗	✓



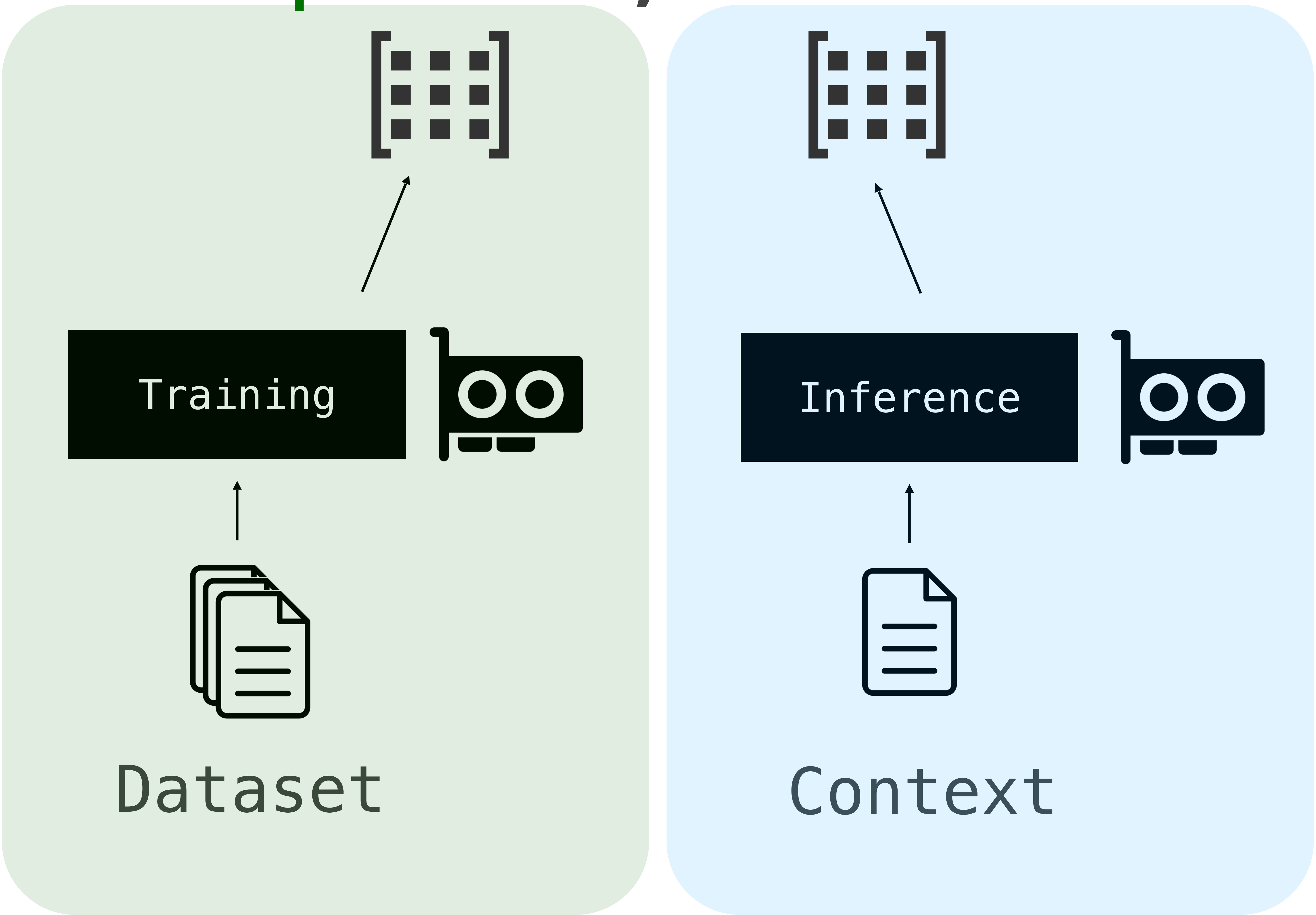
Classic RNNs are **not** GPU-friendly



Why did transformers **win**?

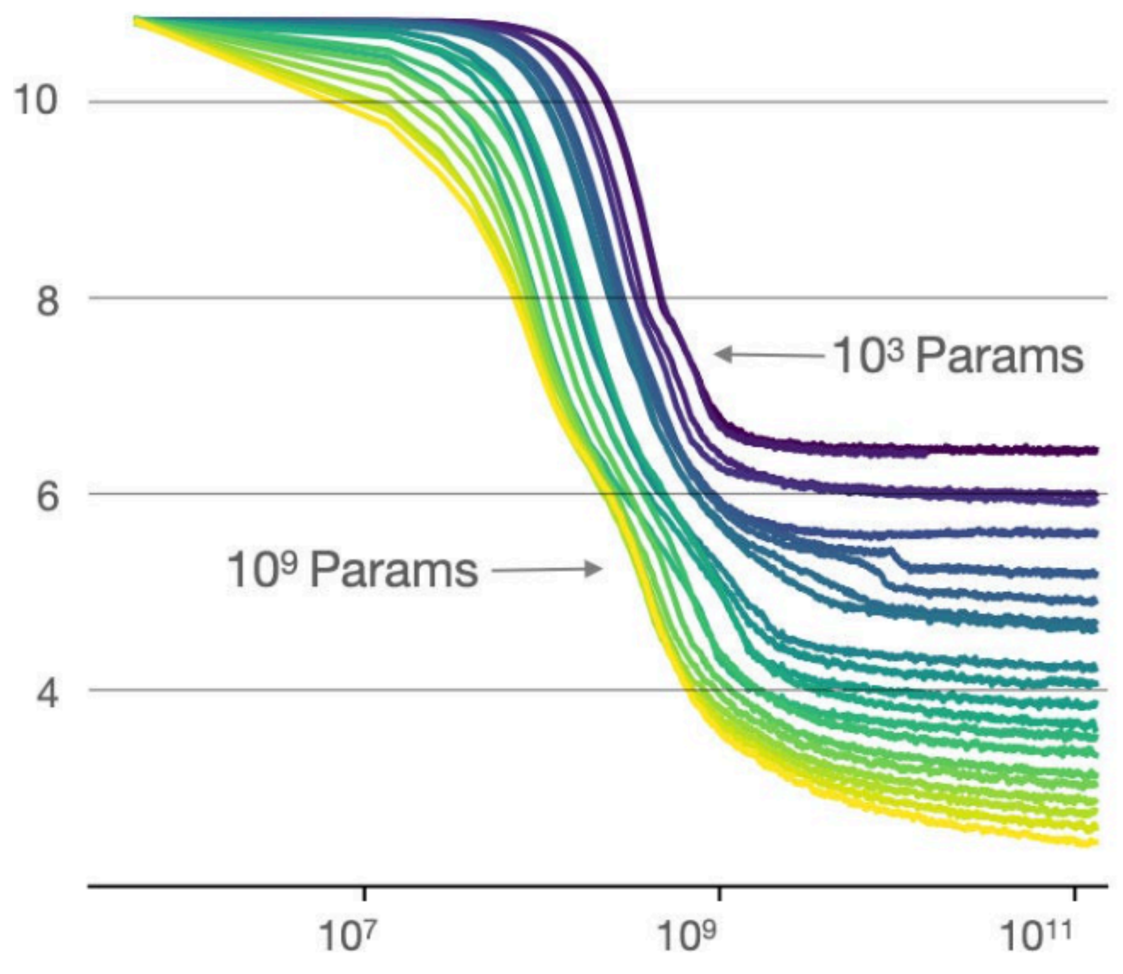
- GPU-friendly
- State gets large

NN(params, state) -> response

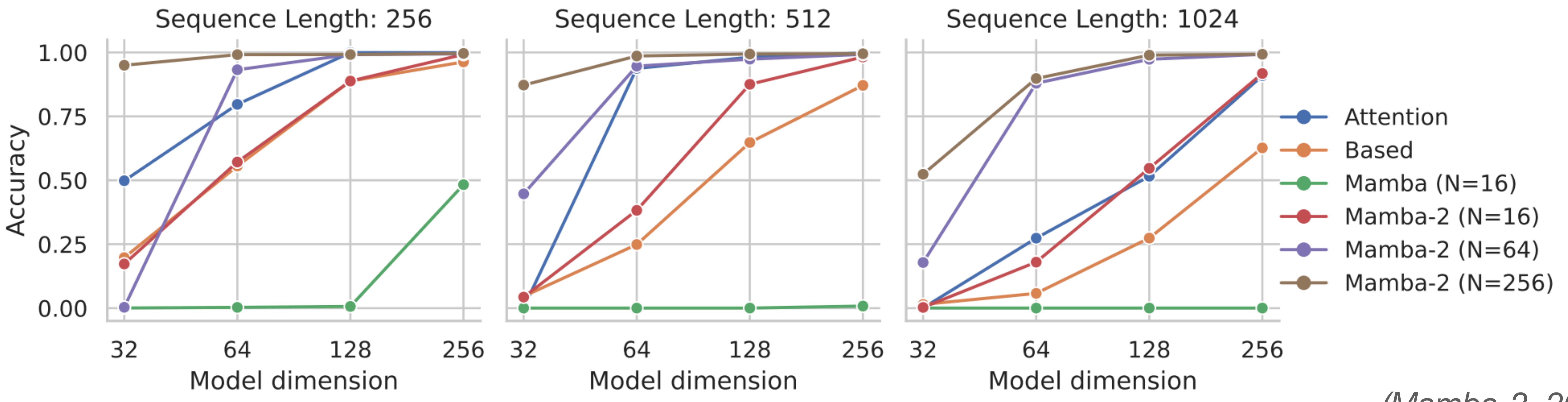


Attention state is **KV cache**

Parameter scaling is well understood

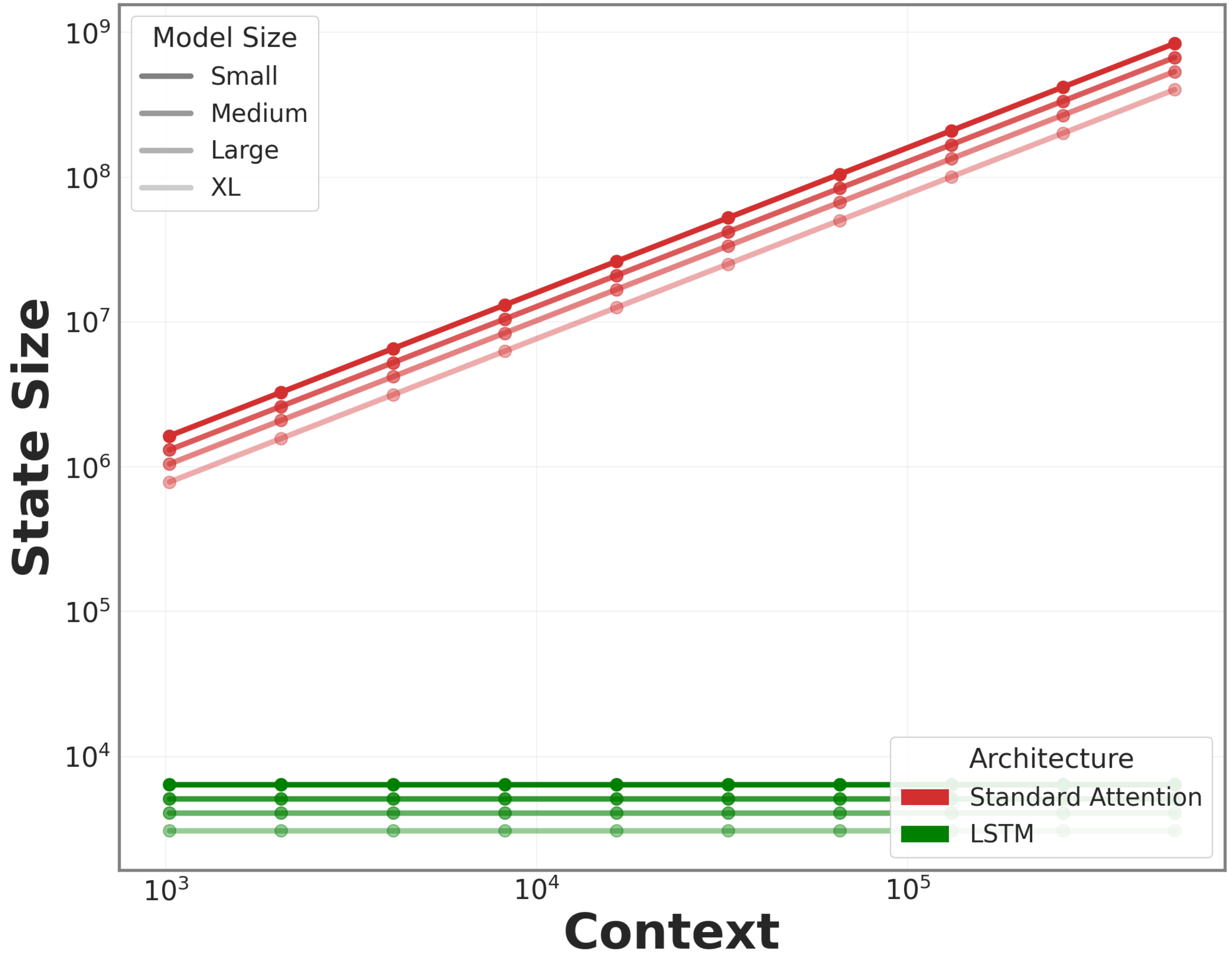


State scaling works the same



(Mamba-2, 2023)

Transformers have far larger states:



LSTM	**O(ld)**
Attention	**O(ldt)**

Why did transformers win?

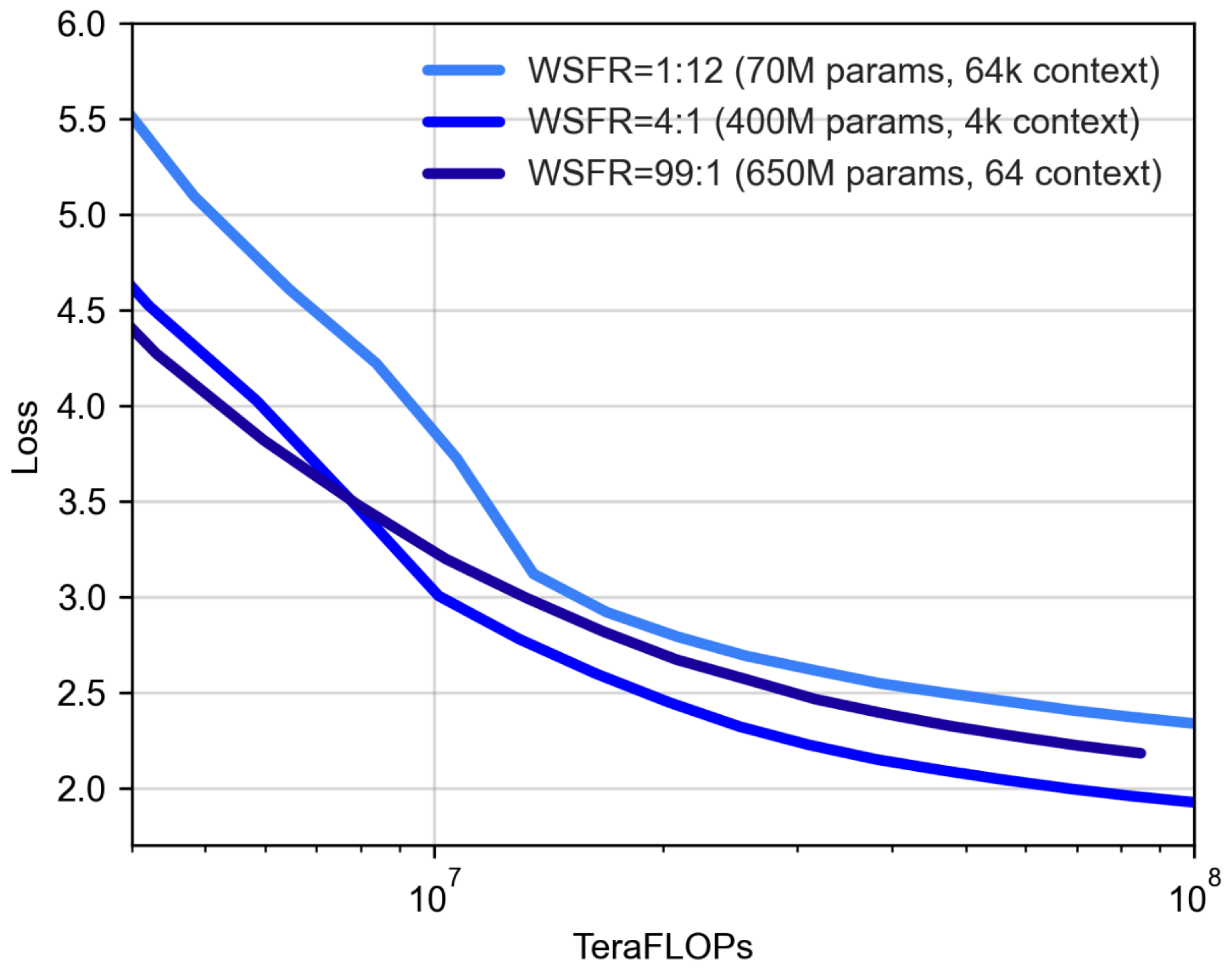
- GPU–friendly
- State gets large

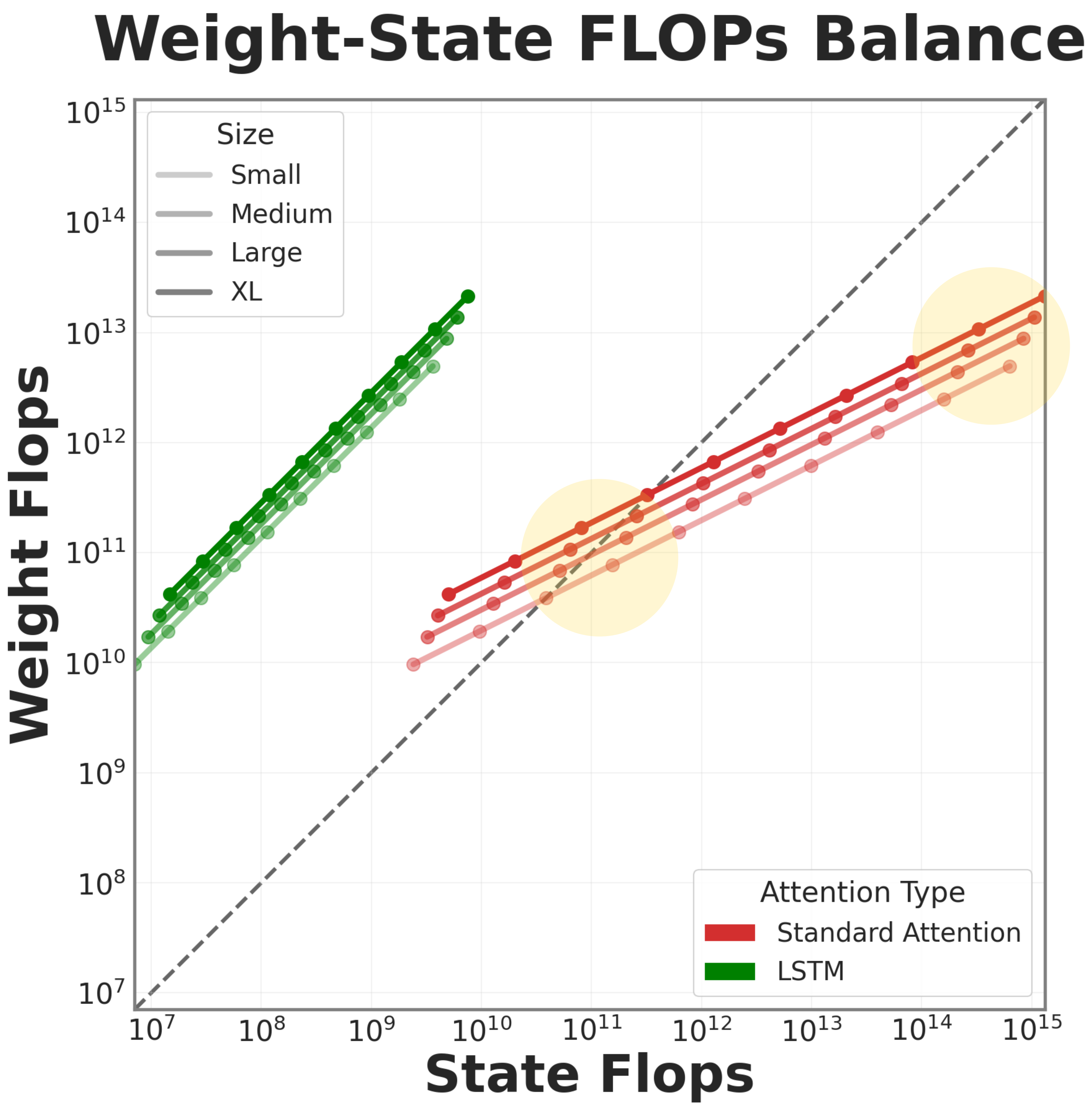
Why will transformers **lose**?

- State gets **too** large

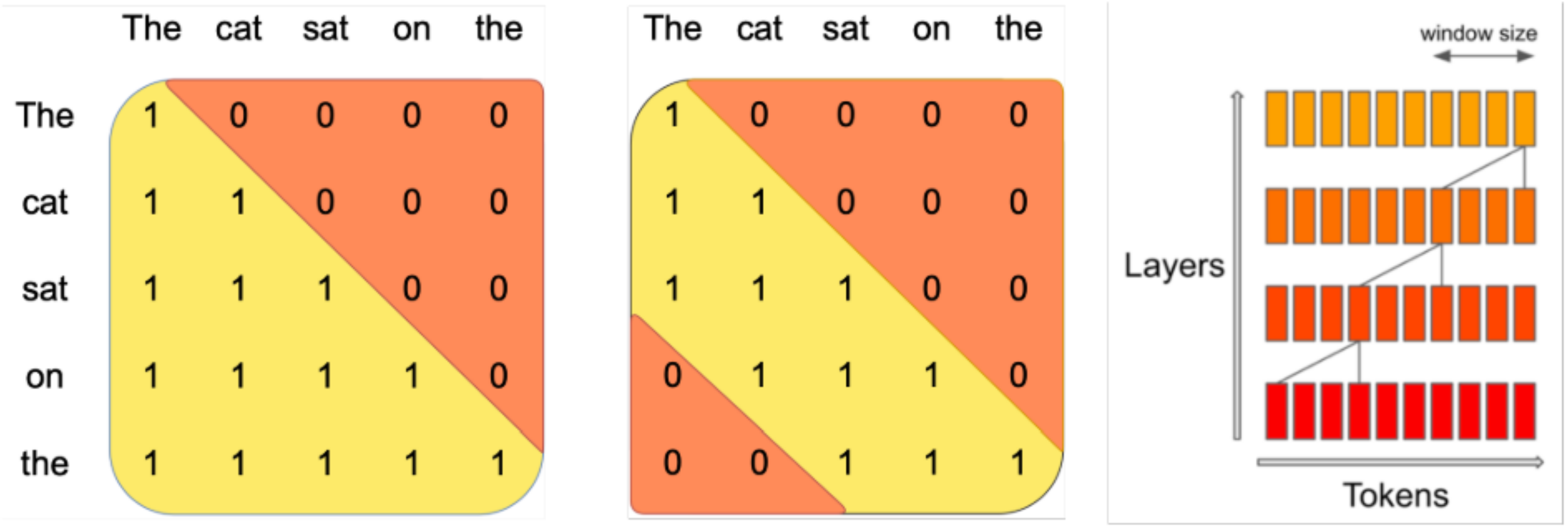
…when we train at long context.

Weight-state FLOP ratio (WSFR) should be balanced!





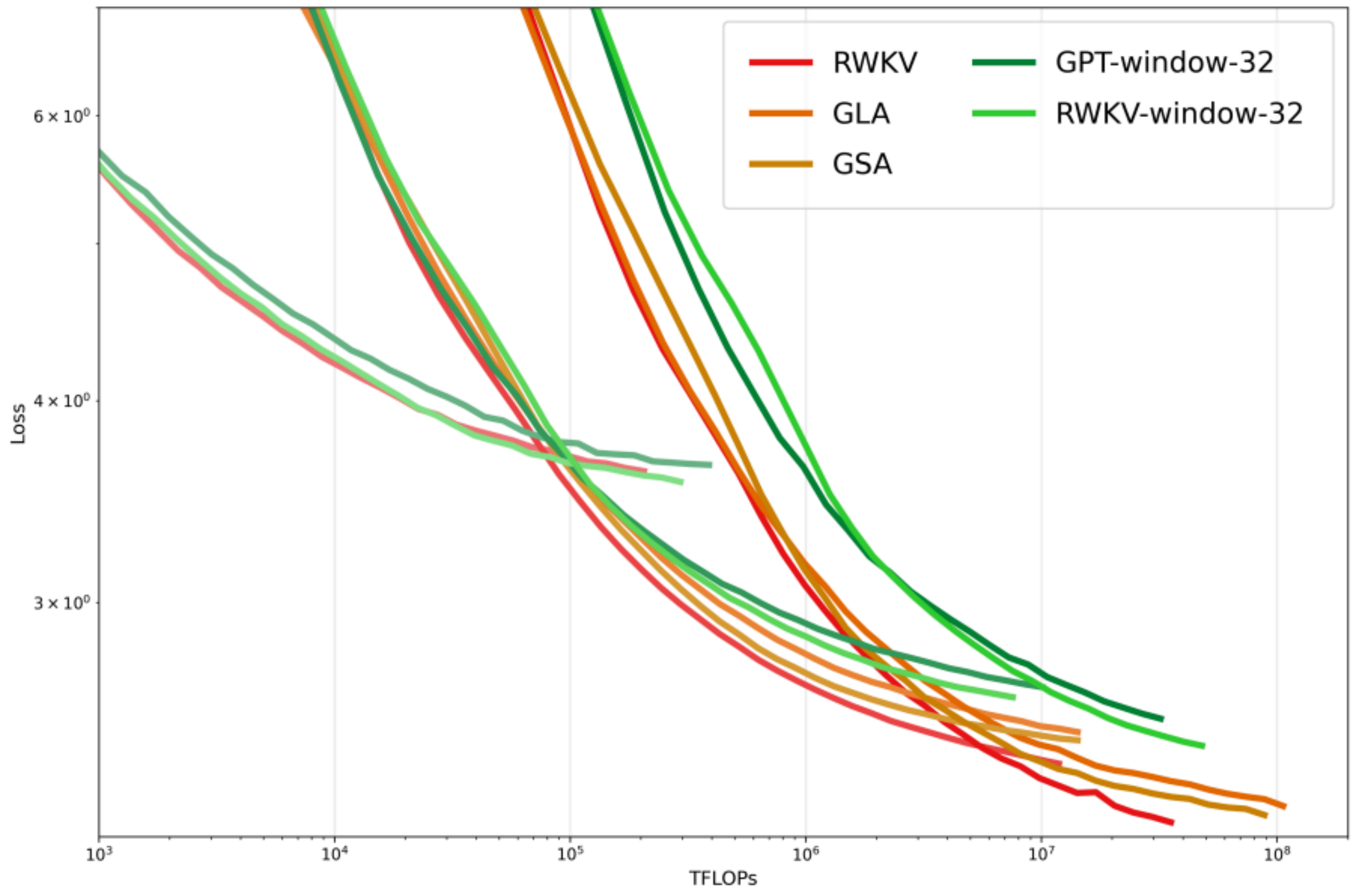
Sliding window attention seems to fix balance



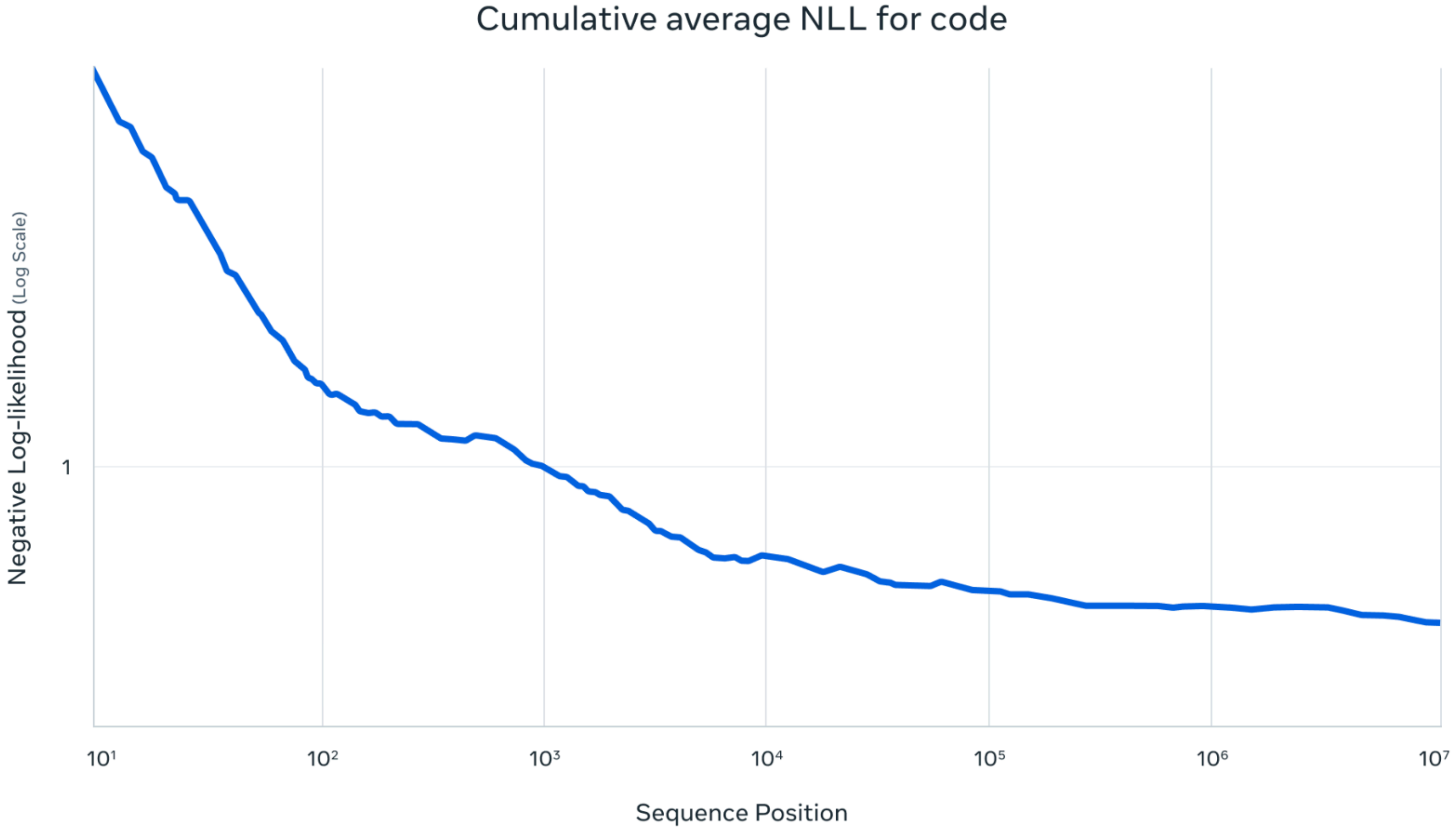
Vanilla Attention **Sliding Window Attention** **Effective Context Length**

(Mistral et al 2023)

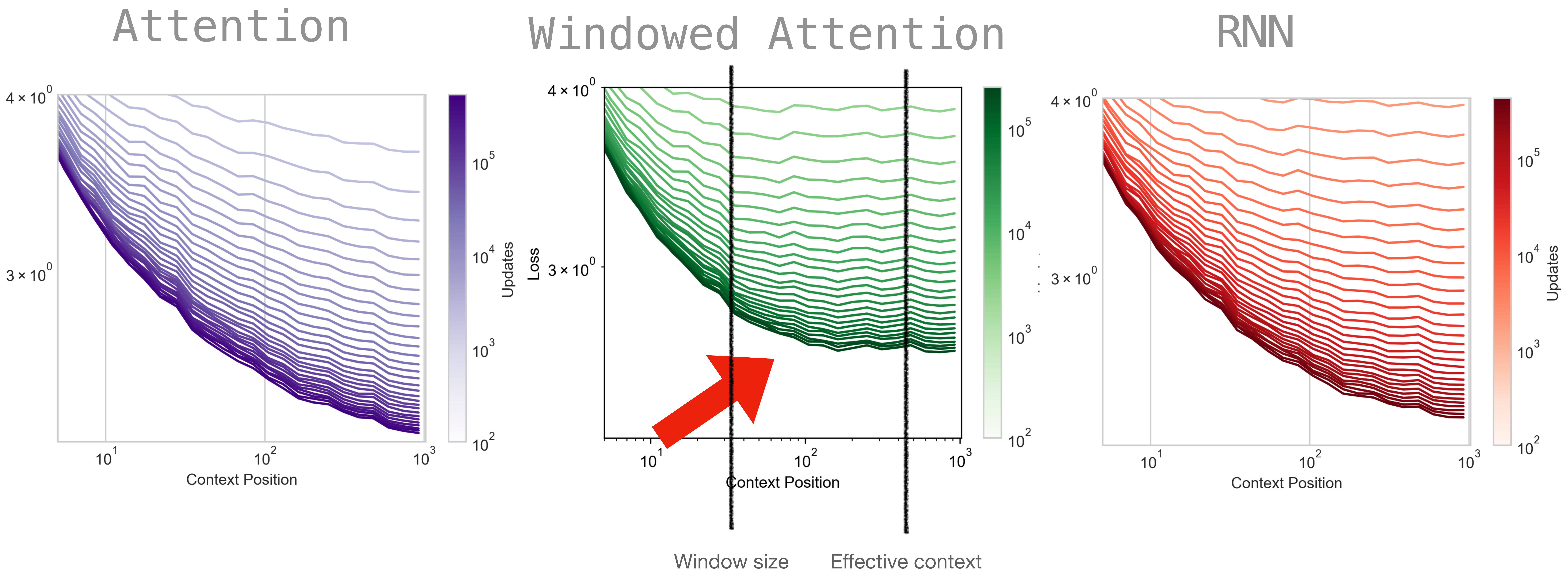
…but windowed attention performs worse than an RNN at equivalent state sizes



In-context learning curve on negative log likelihood:



Llama Team et al, 2025


Attention
Windowed Attention
RNN
4 × 10^0
3 × 10^0
Loss
Context Position
Updates
10^1
10^2
10^3
10^4
10^5
Window size
Effective context

Other ways to fix balance of transformer?

State shape: [layers, time, heads, features]

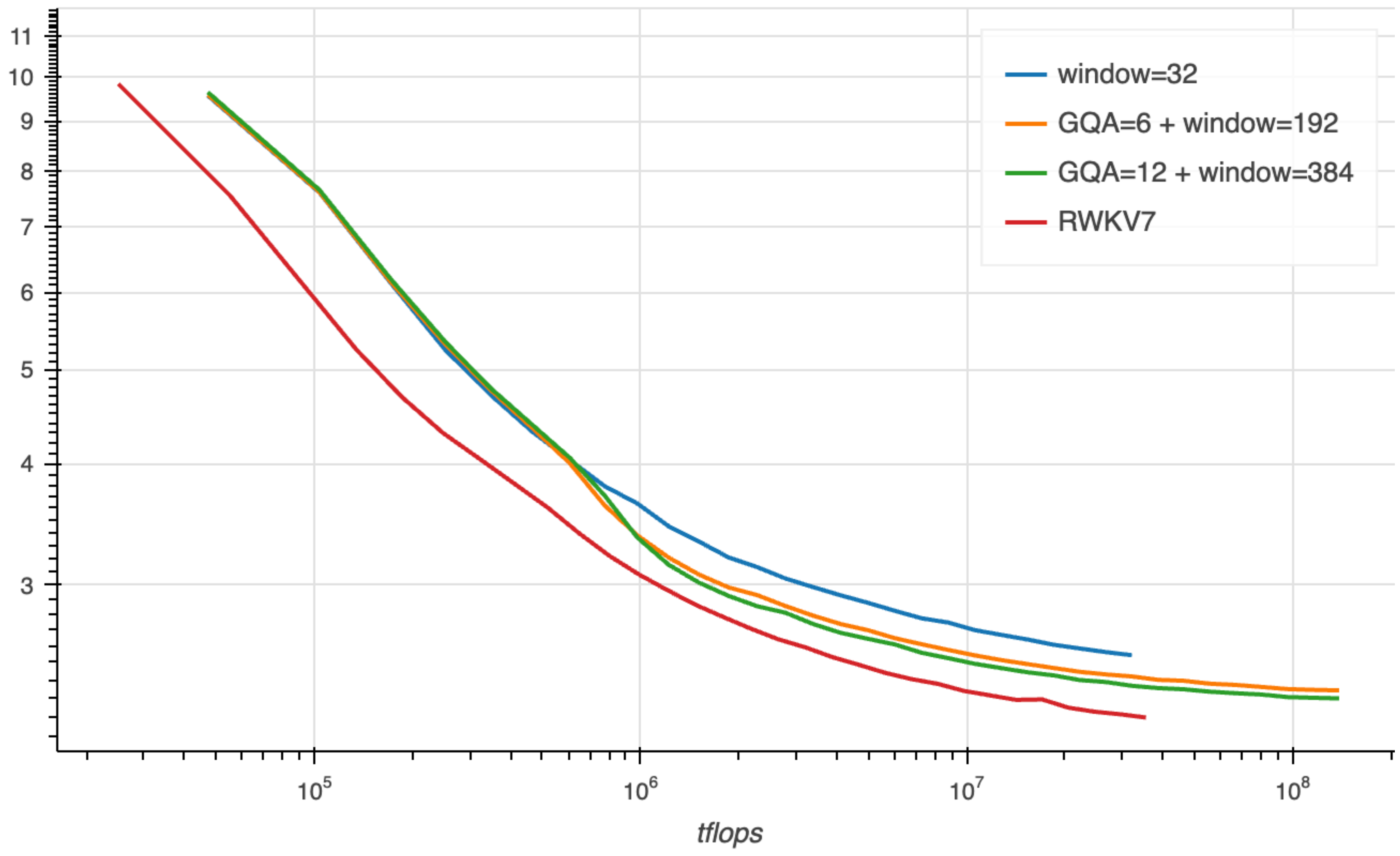
hybrid shrinks this

windowed shrinks this

gqa shrinks this

latent attention shrinks this

RNNs seem to outperform regardless



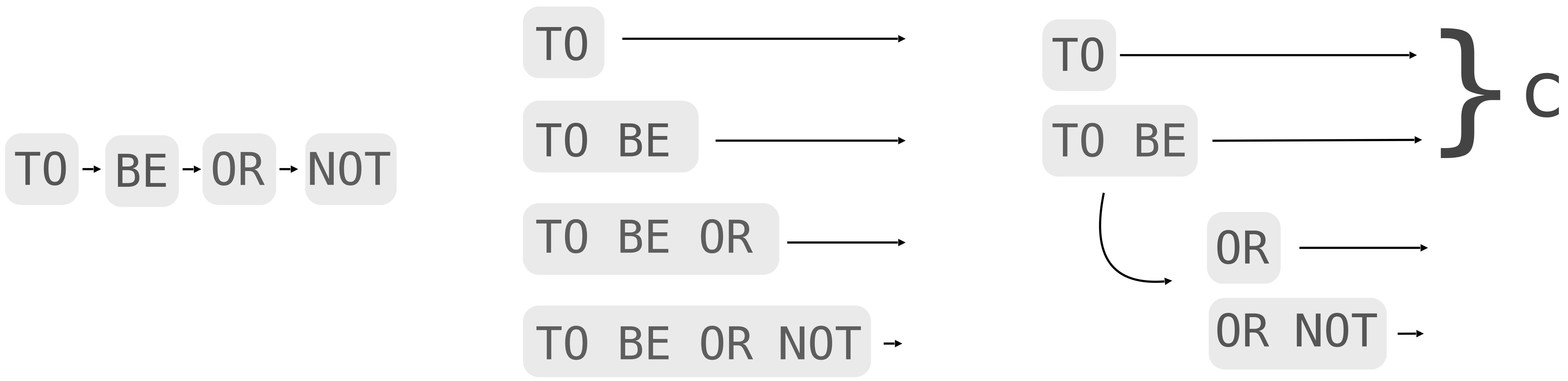
Can we find an RNN that is

- GPU–friendly?
- Large state?

Can we find an RNN that is

- GPU-friendly?
- Large state?

	Classic RNN (e.g. LSTM, GRU)	**Attention** (e.g. Transformer)	**Modern RNN** (e.g. Mamba, RWKV, RetNet)
FLOPs	t	$t^2/2$	$tc/2$
Big matmuls?	✗	✓	✓



From attention

$$\text{attn}_{\text{exp}}(Q, K, V) = \left(\exp(QK^T) \odot M\right) V$$

to *linear* attention

$$\text{attn}^{\phi}_{\text{lin}}(Q, K, V) = \left(\phi(Q)\phi(K)^T \odot M\right) V$$

with state embedding

$$\phi : \mathbb{R}^d \to \mathbb{R}^D$$

Linear attention

$$\text{attn}_{\text{lin}}^{\phi}(Q, K, V) = \left(\phi(Q)\phi(K)^T \odot M\right) V$$

has an equivalent recurrent form

$$\text{attn}_{\text{lin}}^{\phi}(Q, K, V)_i = S_i \phi(Q_i) \qquad S_i = S_{i-1} + V_i \phi(K_i)^T$$

Attention form + recurrent form -> chunk-wise form

output

attention on *c* elements

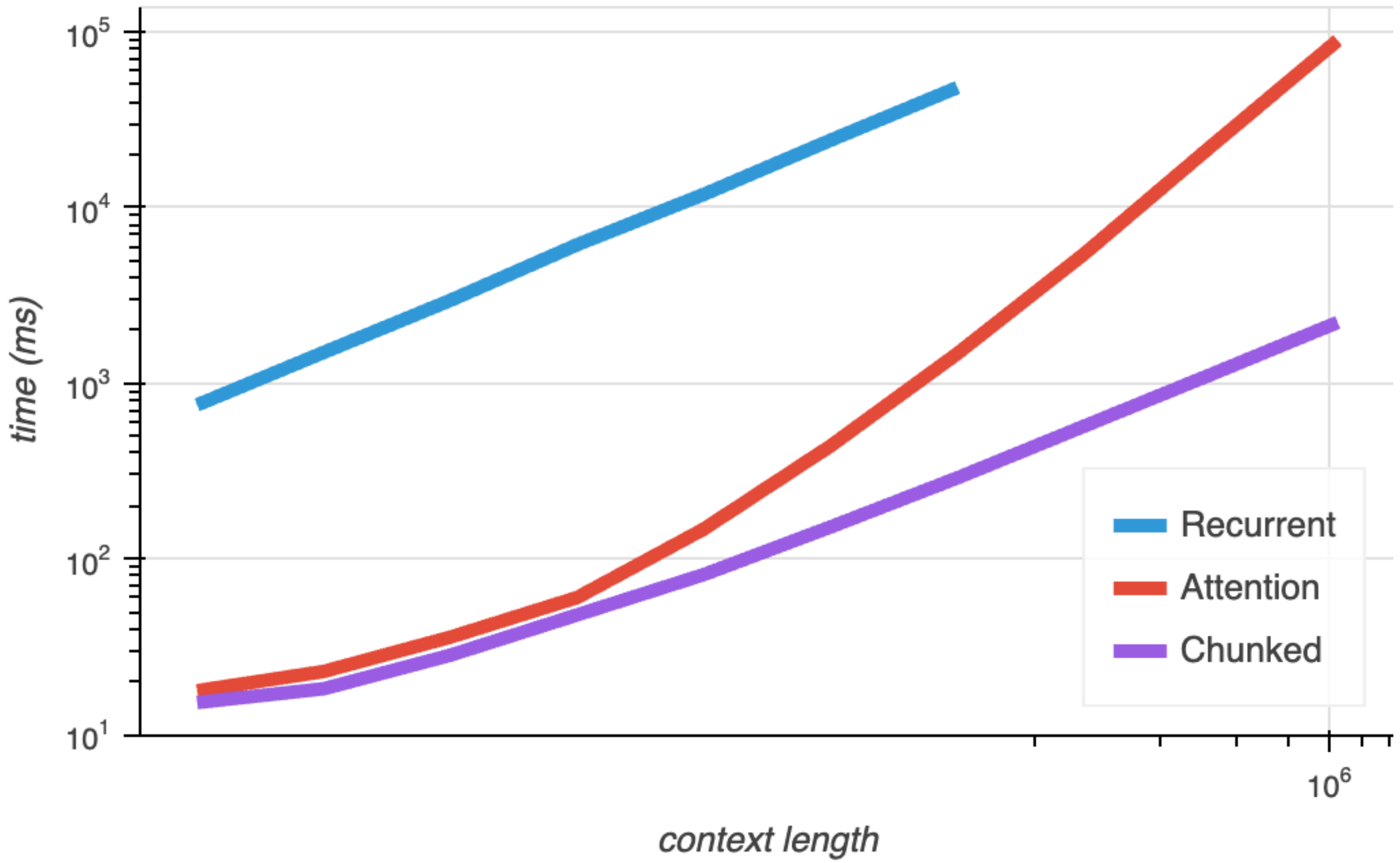
$$Y_{(i)_c} = S_{ci} Q_{(i)_c} + V_{(i)_c} \left(Q_{(i)_c} K^T_{(i)_c} \odot M \right)$$

influence from past

$$S_{c(i+1)} = S_{ci} + V_{(i)_c} K^T_{(i)_c}$$

influence on future

Chunk-wise, RNNs **are** GPU-friendly!



Can we find an RNN that is

- GPU-friendly? ✔
- Large state?

Sliding windowed attention *shrinks a KV cache*.

What if instead we *enlarge an RNN state*?

$$\phi : \mathbb{R}^d \to \mathbb{R}^D$$

From attention

$$\text{attn}_{\text{exp}}(Q, K, V) = \left(\exp(QK^T) \odot M\right) V$$

to *power attention*

$$\text{attn}^{p}_{\text{pow}}(Q, K, V) = \left((QK^T)^p \odot M\right) V$$

Let

$$\phi = \text{TPOW}_p(x) = \begin{bmatrix} x_1 \cdots x_1 \\ x_1 \cdots x_2 \\ \vdots \\ x_d \cdots x_d \end{bmatrix} = \begin{bmatrix} \vdots \\ \prod_k x_{i_k} \\ \vdots \end{bmatrix}_{(i_1,\cdots,i_p)\in\mathbb{N}_d^{\times p}}$$

(outer product of x with itself, p times)

Then: $$\phi(Q_i)^T\phi(K_j) = (Q_i^T K_j)^p$$

…so power attention is linear attention!

$$\text{attn}_{\text{pow}}^p(Q,K,V) = ((QK^T)^p \odot M)\,V = (\phi(Q)\phi(K)^T \odot M)\,V = \text{attn}_{\text{lin}}^{\phi}(Q,K,V)$$

We can find even better embeddings!

TPOW produces a symmetric tensor

$$x = [a, b, c]$$

$$\text{TPOW}_2(x) = [aa, ab, ac \\ ab, bb, bc \\ ac, bc, cc]$$

SPOW produces *unique elements* of that tensor…

$$\text{SPOW}_2(x) = [aa, ab, ac, bb, bc, cc]$$

…scaled by coefficients based on the count:

$$\text{SPOW}_2\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 x_1 \\ \sqrt{2}\, x_1 x_2 \\ x_2 x_2 \end{bmatrix} \qquad \text{SPOW}_3\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 x_1 x_1 \\ \sqrt{3}\, x_1 x_1 x_2 \\ \sqrt{3}\, x_1 x_2 x_2 \\ x_2 x_2 x_2 \end{bmatrix}$$

which gives:

1. *The dimensionality D is given by $\binom{d+p-1}{p}$ (the binomial n choose k)*
2. *The inner products* $\text{SPOW}_p(q)^T \text{SPOW}_p(k) = (q^T k)^p$

p	TPOW D	SPOW D	Savings
2	4096	2080	49%
3	262144	45760	82%
4	16777216	766480	95%
5	1073741824	10424128	99%
6	68719476736	119877472	99.8%

Recap:

Linear attention + ϕ = power-p attention

Can be computed chunk-wise in O(t) FLOPs

State size can be expanded *independent of params* with p:

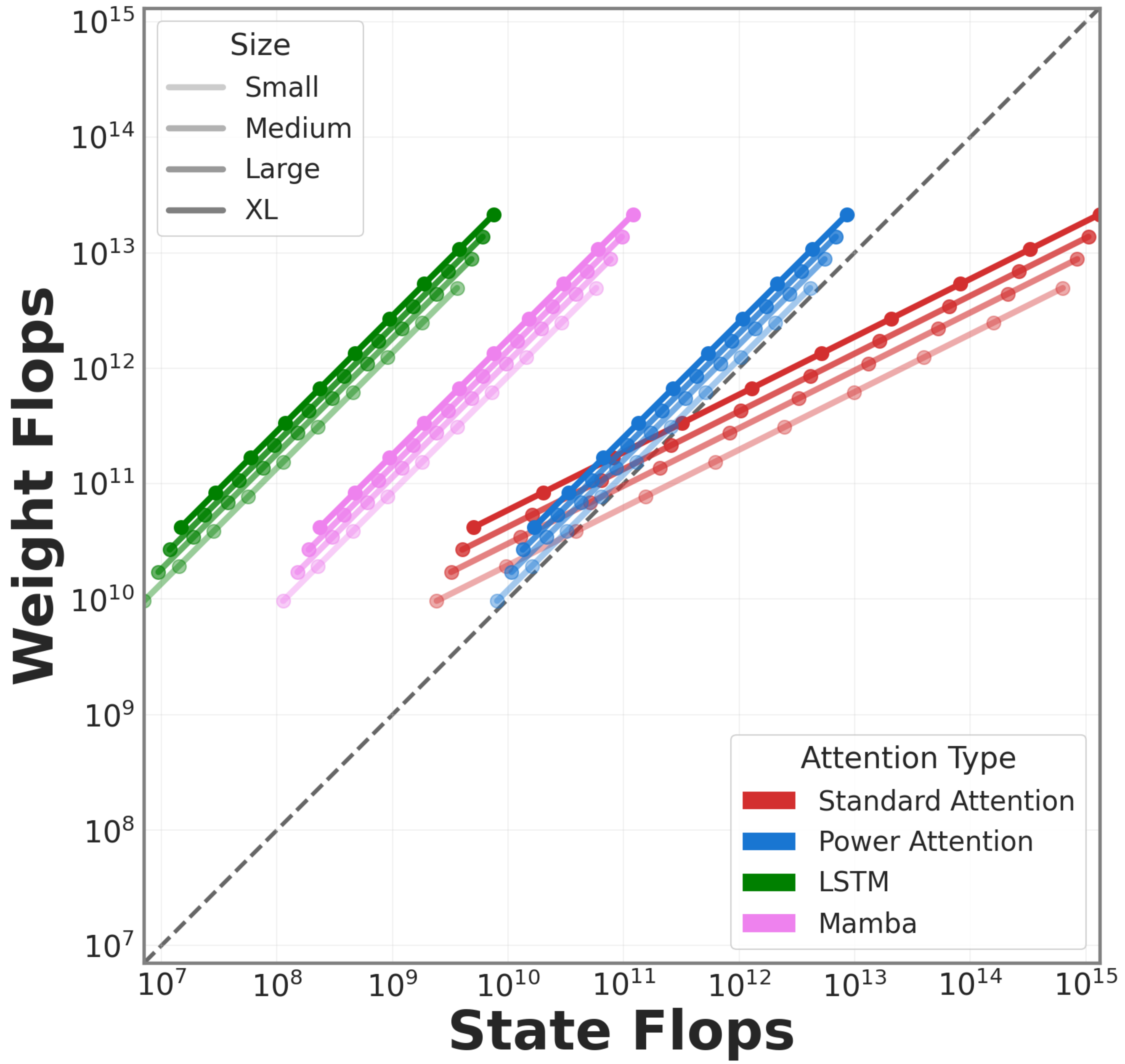
p	SPOW D
2	2080
3	45760
4	766480
5	10424128
6	119877472

Can we find an RNN that is

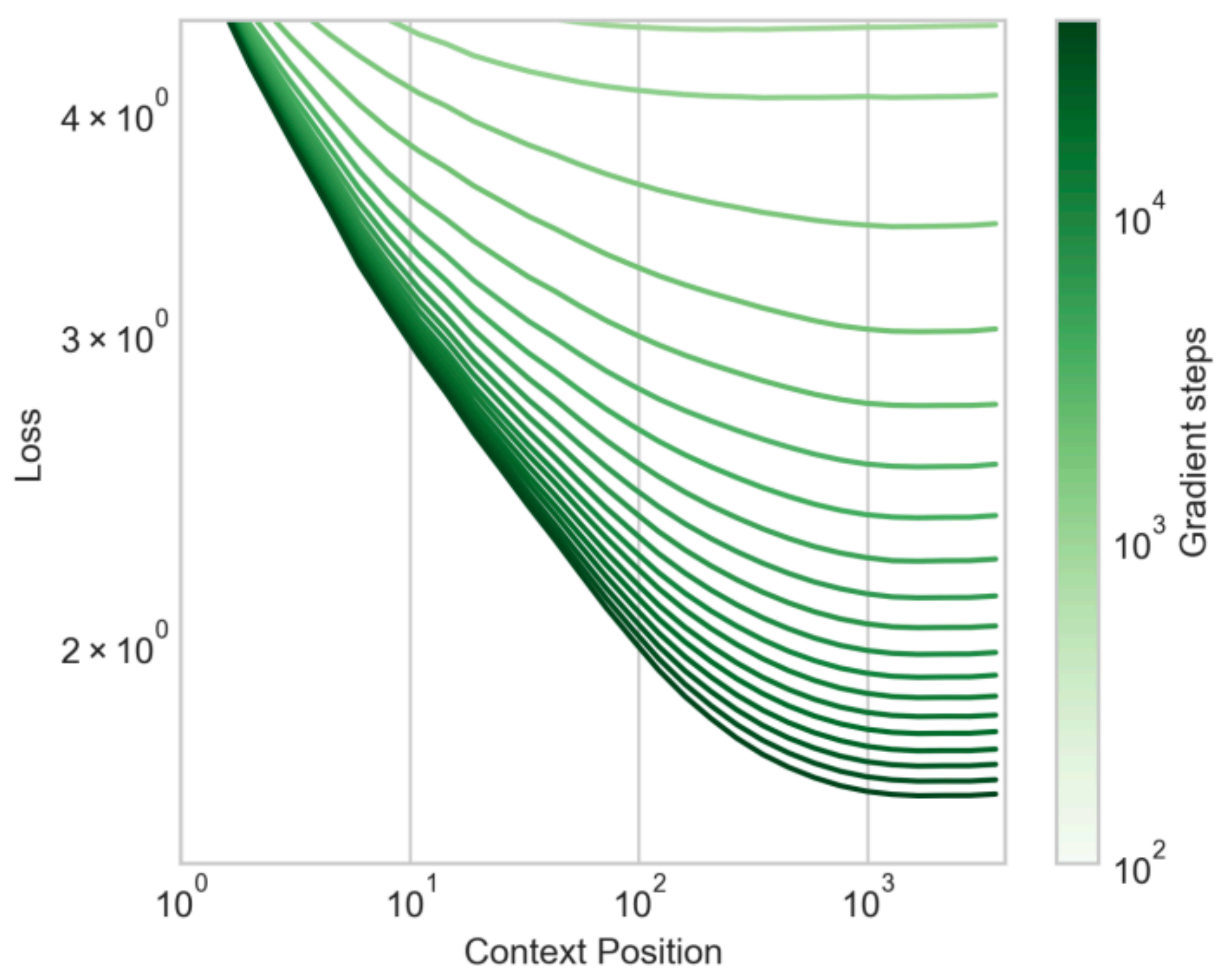
- GPU-friendly? ✓
- Large state? ✓

Time to see power attention in action…

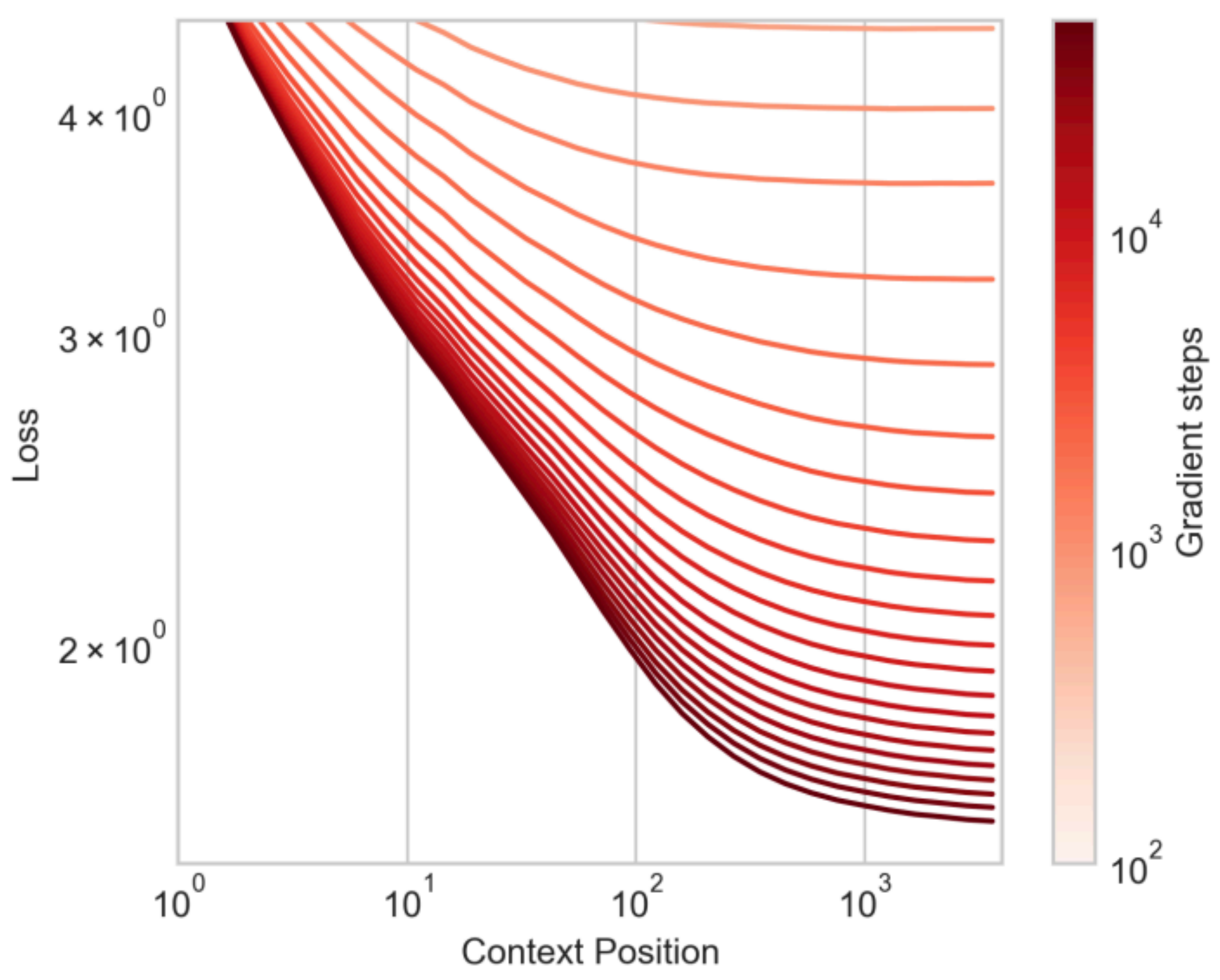
Power attention balances the WSFR



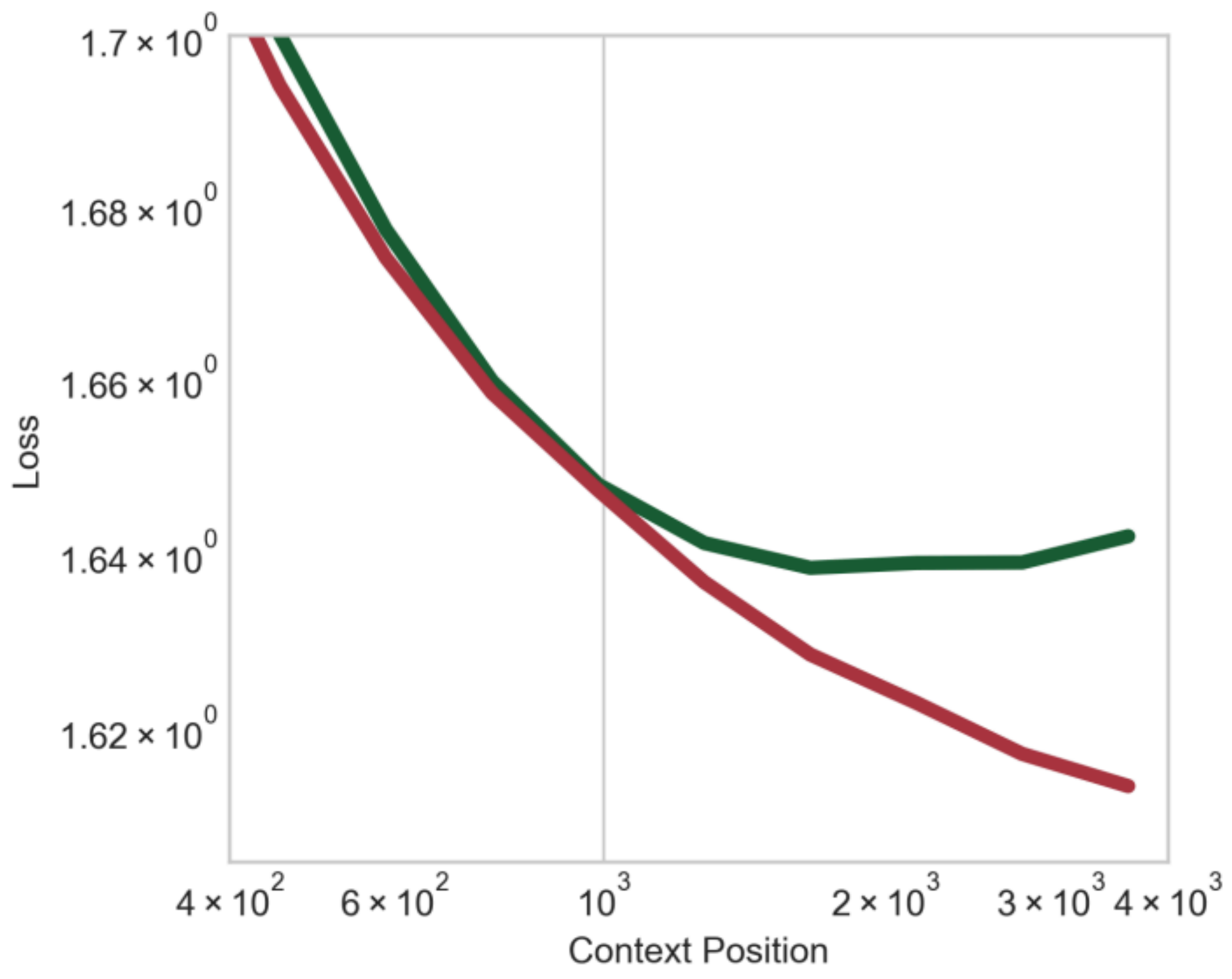
Power attention in-context learning is better than equivalent windowed attention



(a) Window-1k attention.

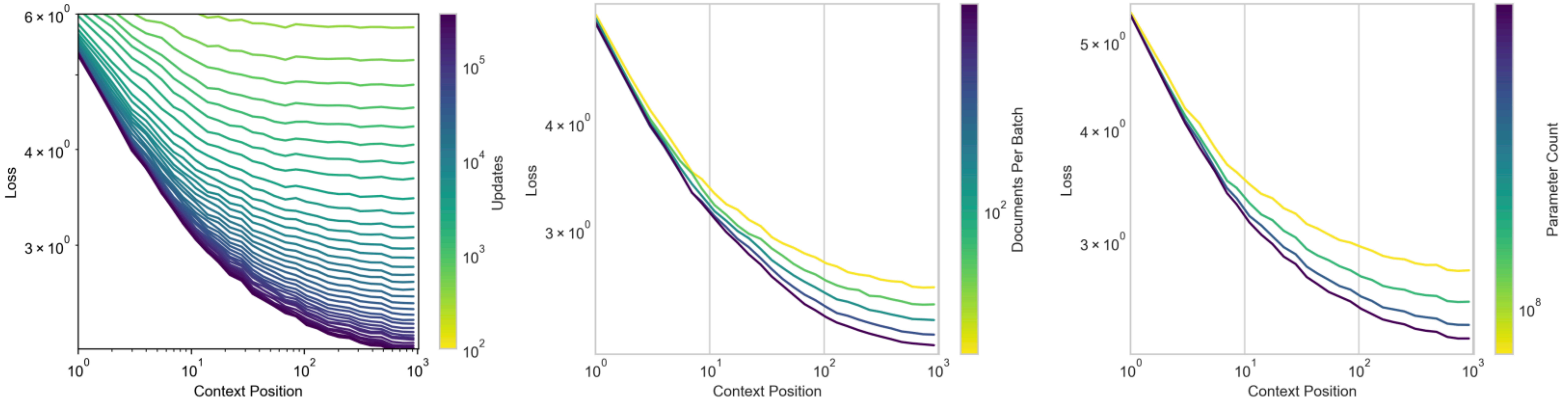


(b) $p = 2$ power attention.



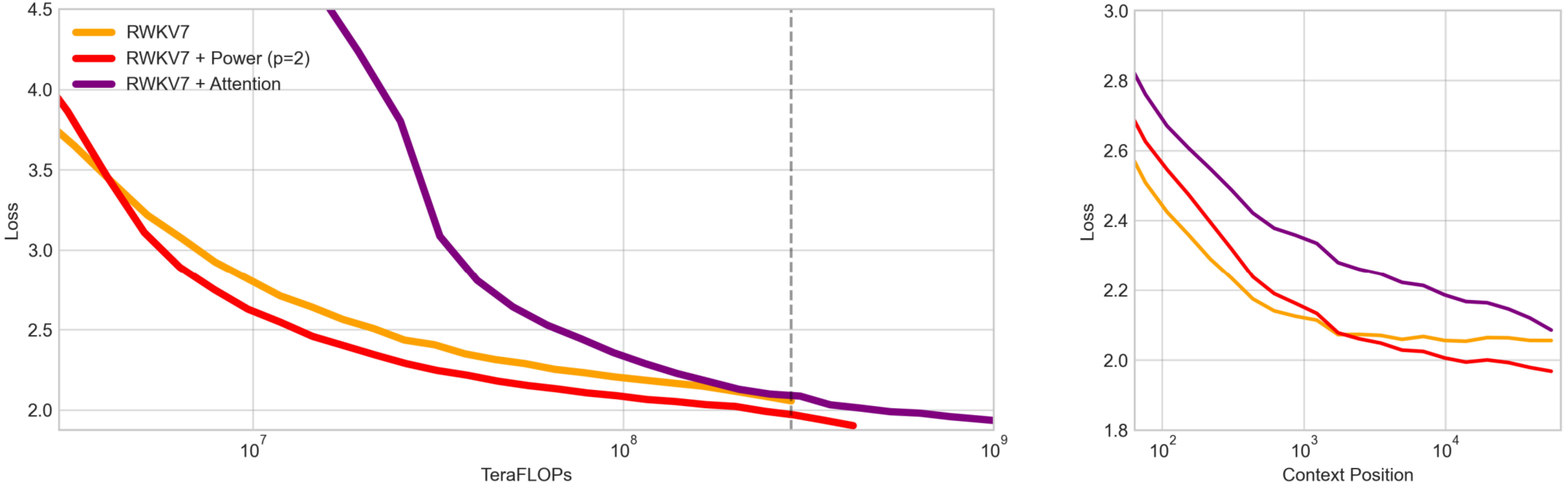
(c) Close-up on ICL curves at 50k.

Power attention scales with conventional axes



(a) Gradient updates. (b) Documents per batch. (c) Parameter count.

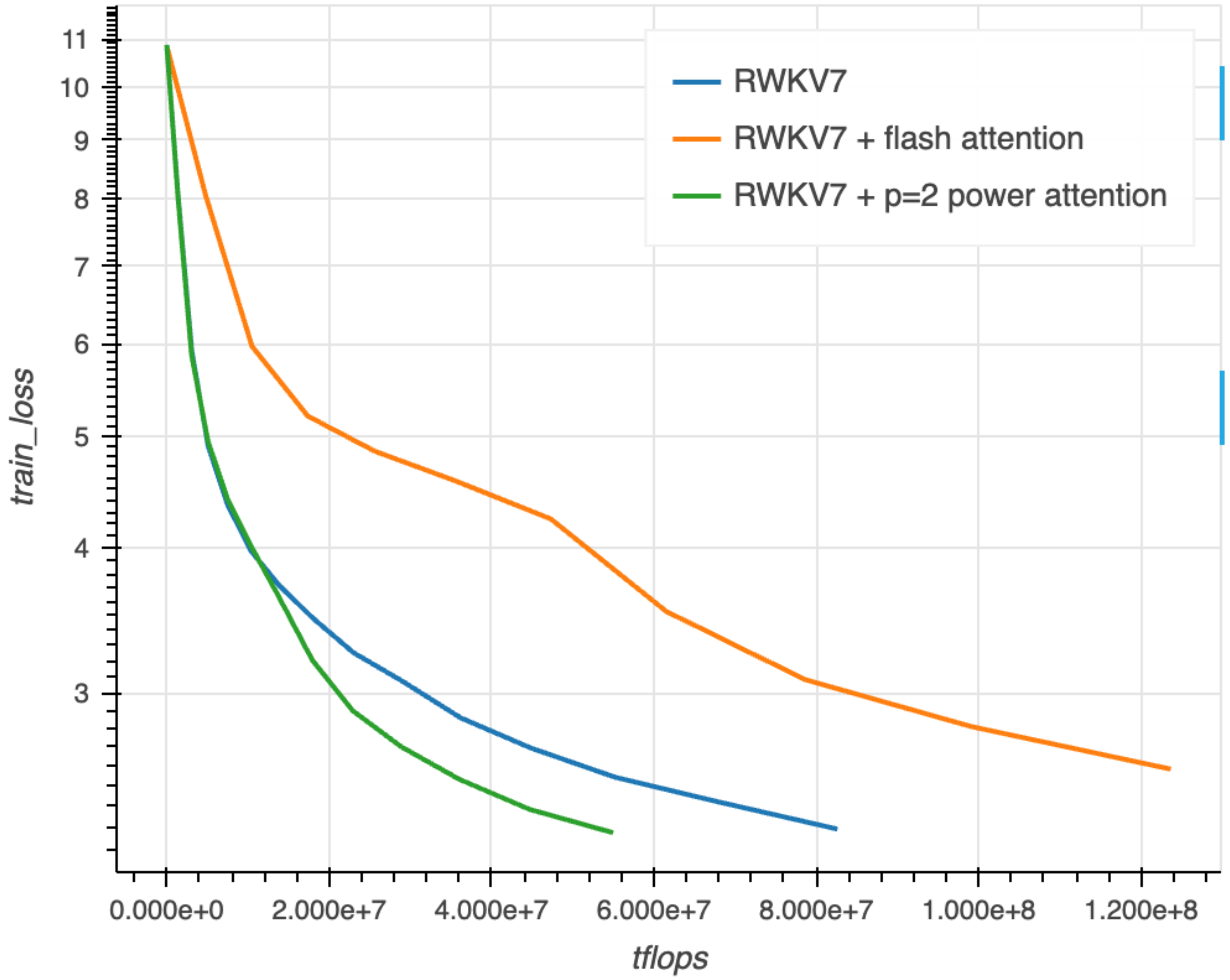
Power attention dominates on long-context training



(a) Heldout best-context loss across training.

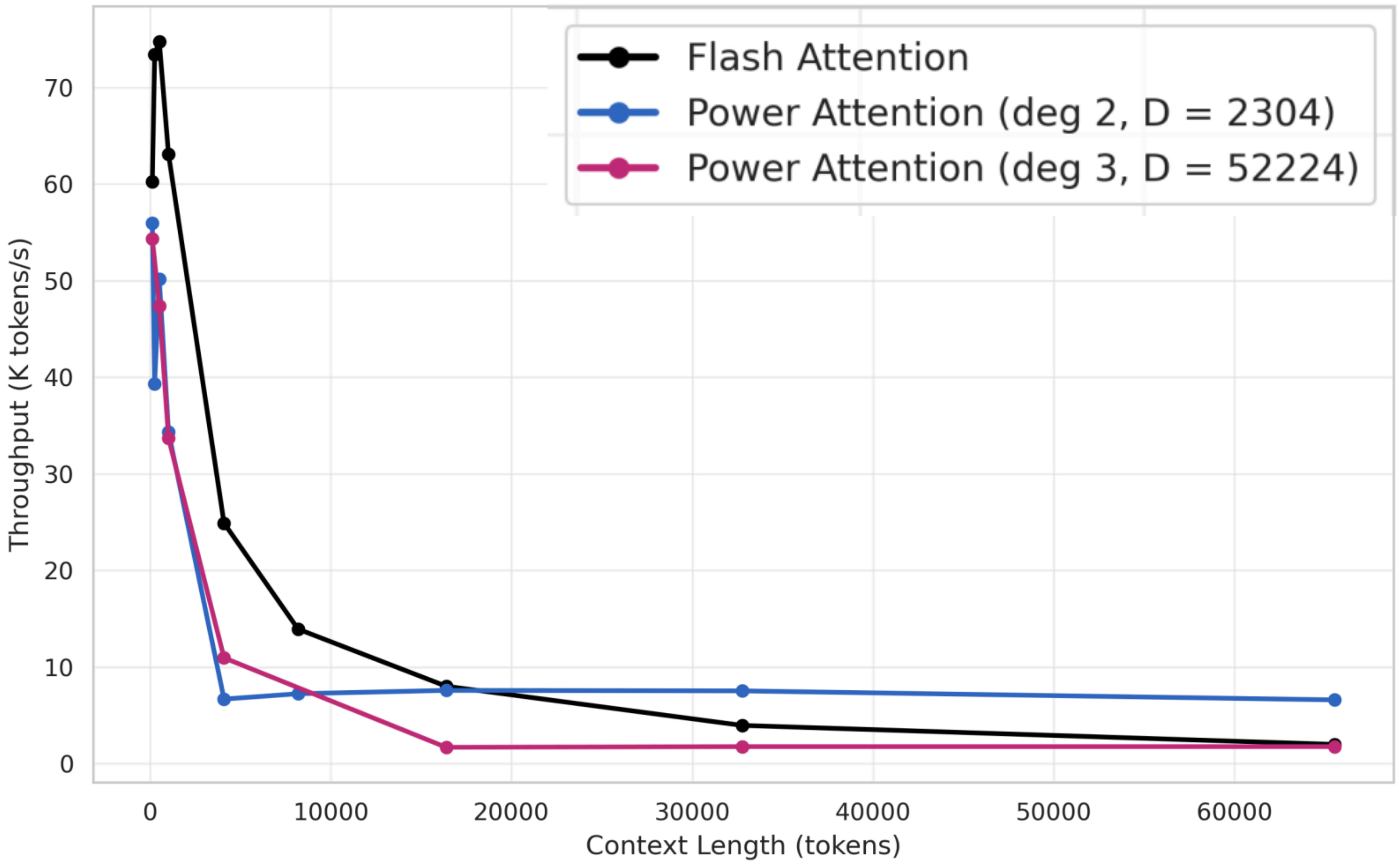
(b) ICL after 3e8 TeraFLOPs.

Trend holds at scale (1.5B parameters, 32k context)



manifest ai

Hardware-aware kernels available open-source:
https://github.com/m-a-n-i-f-e-s-t/power-attention



FLA pull request
coming soon!

Many thanks to

The San Francisco Compute Company

for supporting our work

contact: jacob@manifestai.com

https://github.com/m-a-n-i-f-e-s-t/power-attention