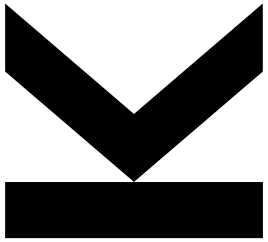
NXAI JKU

JOHANNES KEPLER
UNIVERSITÄT LINZ

xLSTM Scaling Laws: Competitive Performance with Linear Time-Complexity



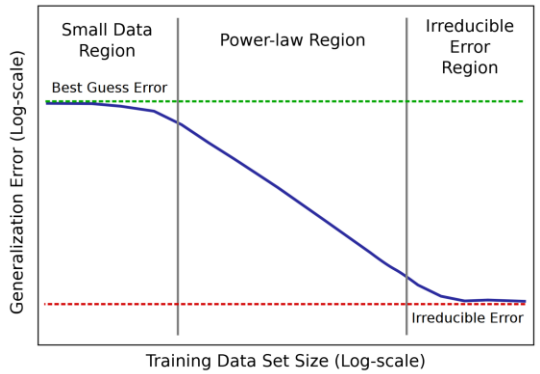
ICLR 2026

Maximilian Beck, Kajetan Schweighofer,
Sebastian Böck, Sebastian Lehner, Sepp Hochreiter

March 2026

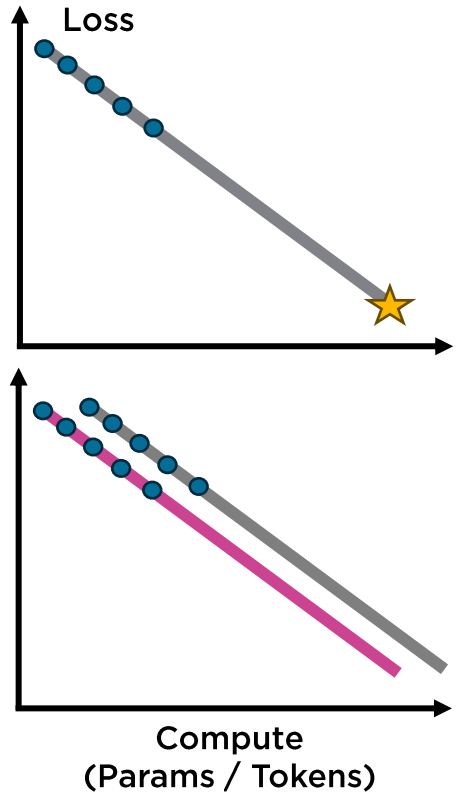
Motivation: Neural Scaling Laws

- Empirical relation between loss and compute, model or dataset size
- These relations follow power law trends,
  and often hold over many orders of magnitude [1,2]



Significant practical implications:

- Predict the outcome of & derisk very large training runs
- Compare model classes
- Allow for compute-optimal allocation of model parameters N & training tokens D [2]



Previous neural scaling law studies [2,3, ...] have focused exclusively on the Transformer architecture [4]

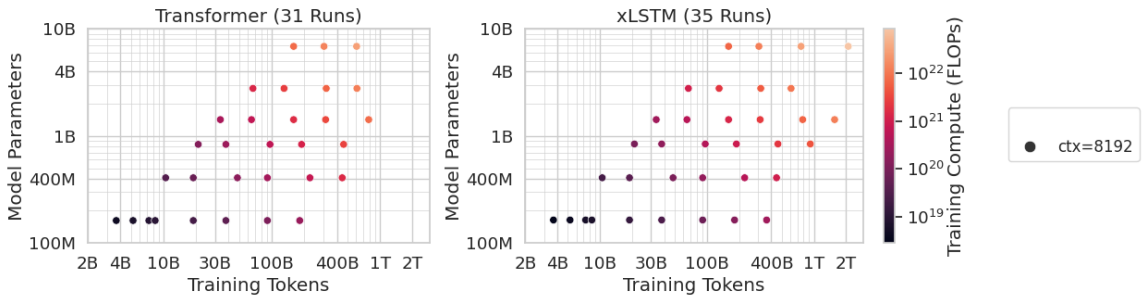
<u>This paper:</u>
Large scale scaling law comparison of xLSTM [5] with Transformers

[1] Hestness, Narang, Ardalani, ..., Zhou (2017) Deep Learning Scaling is Predictable, Empirically, ArXiv
[2] Kaplan, McCandlish, Henighan, ..., Amodei (2020) Scaling Laws for Neural Language Models, ArXiv
[3] Hoffmann, Borgeaud, Mensch, ..., Sifre (2022) Training Compute-Optimal Large Language Models, ArXiv
[4] Vaswani, Shazeer, Parmer, ..., Polosukhin (2017) Attention Is All You Need, NeurIPS 17
[5] Beck, Pöppel, Spanring, ..., Hochreiter (2024) xLSTM: Extended Long Short-Term Memory, NeurIPS 24

Outline of our scaling law study

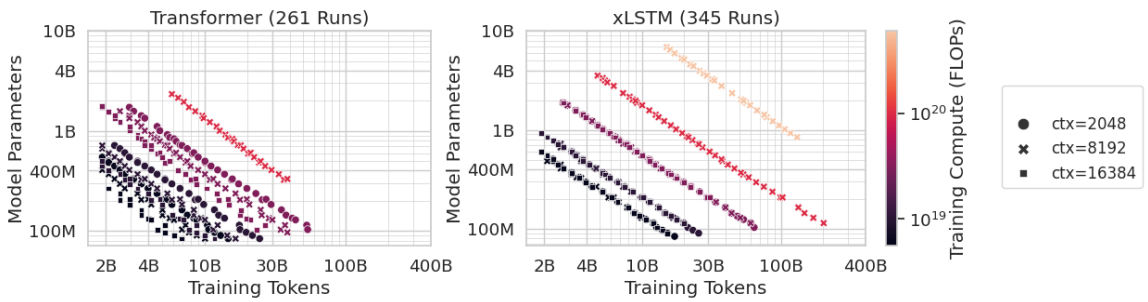
Token/param

Fix model parameters,
vary training tokens



IsoFLOP

Fix compute budget,
vary model parameters & training tokens



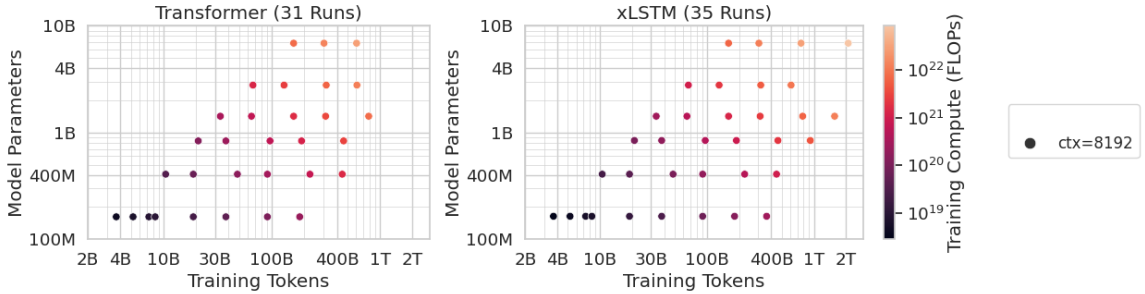
Inference speed

We compare **prefill and generation speed**

Outline of our scaling law study

Token/param

Fix model parameters,
vary training tokens



We study:

- Loss vs. compute
- Overtraining regime

IsoFLOP

Fix compute budget,
vary model parameters & training tokens

Inference speed

We compare prefill and generation speed

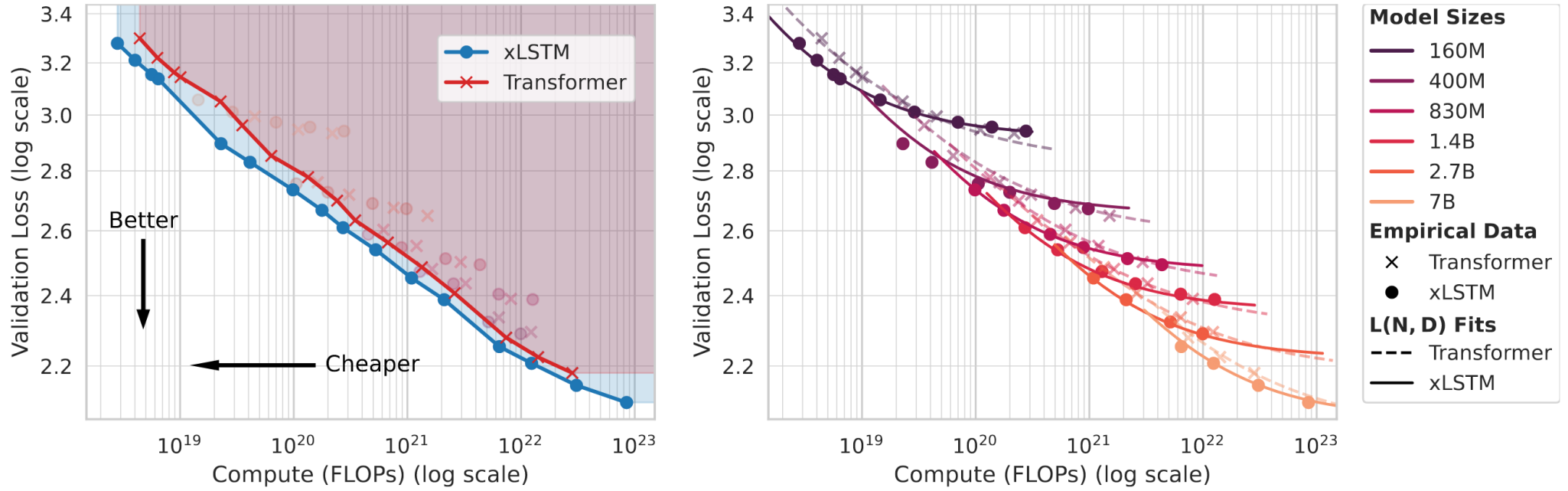
[Token/param] Parametric Fit Approach: Loss vs. compute

- We use token/param runs including non-compute optimal configurations
- Fit the loss as a power-law in model parameters and tokens

$$\hat{L}(N, D) = E + (AN^{-\alpha} + BD^{-\beta})^{\gamma}$$

	Huber δ	$\log A$	$\log B$	$\log E$	α	β	γ
Transformer	10^{-5}	12.96	14.35	0.05	0.58	0.55	0.28
	10^{-3}	11.99	13.35	0.01	0.53	0.51	0.29
	$\geqslant 10^{-1}$	14.45	16.33	0.09	0.64	0.63	0.25
xLSTM	10^{-5}	16.13	17.10	0.07	0.71	0.66	0.24
	10^{-3}	16.22	17.31	0.11	0.73	0.67	0.24
	$\geqslant 10^{-1}$	15.46	16.53	0.18	0.71	0.65	0.26

Fitted parameters for different huber deltas



➡ xLSTMs are Pareto-dominant to Transformers in terms of loss vs. compute

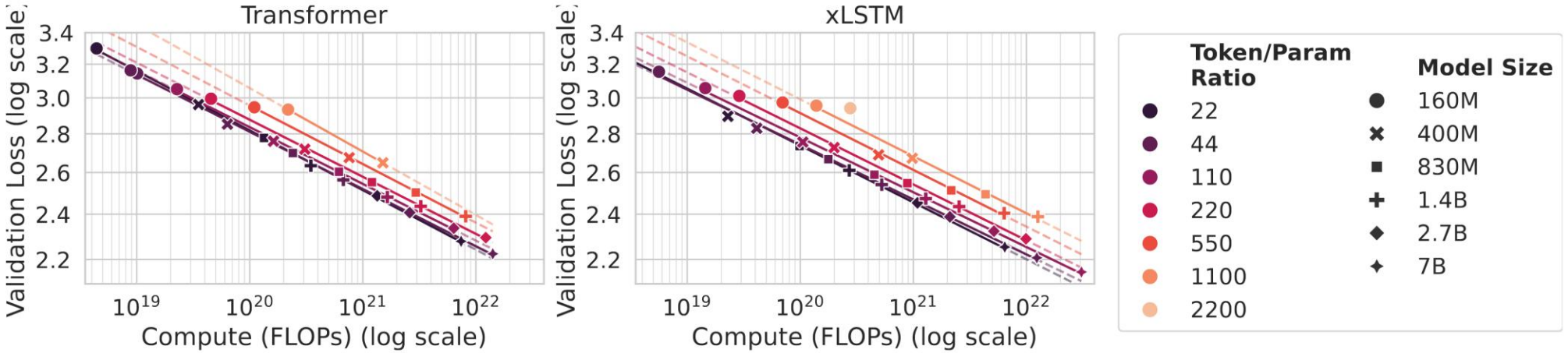
[Token/param] Scaling in the overtraining regime

- For Transformers, optimal token/parameter ratio is ~22 [1]
- Large models are slow during inference
  - Overtraining: training smaller (faster) models longer
  - Transformers scale reliably in overtraining,[2] also xLSTM?

M	Transformer	xLSTM
22	0.050	0.047
44	0.048	0.046
110	0.047	0.046
220	0.048	0.047
550	0.049	0.047
1100	-	0.047

Power law exponents

- Fit power laws of the form $\hat{L}(C) = \lambda \cdot C^{-\eta}$ to runs with different token/param ratios M



➡ **Exponents η of xLSTM remain constant, indicated by the parallel lines in the log-log plots**

➡ **We can expect consistent improvements for xLSTM for very long training**

[1] Hoffmann, Borgeaud, Mensch, ..., Sifre (2022) Training Compute-Optimal Large Language Models, ArXiv
[2] Gadre, Smyrnis, Shankar, ..., Schmidt (2024) Language models scale reliably with over-training and on downstream tasks, ArXiv

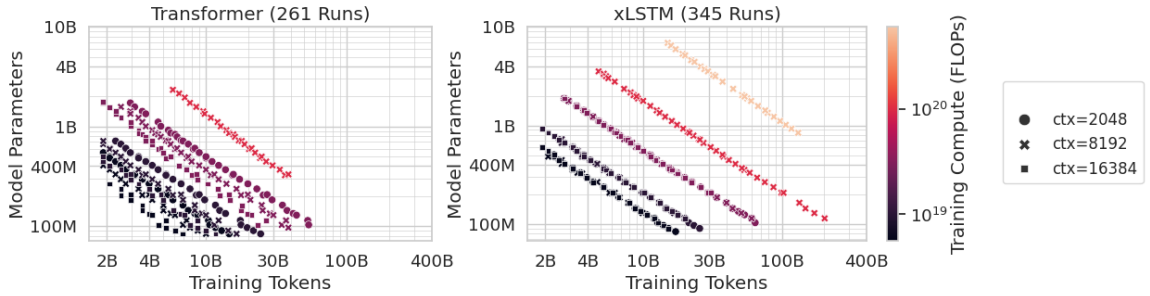
Outline of our scaling law study

Token/param

Fix model parameters,
vary training tokens

IsoFLOP

Fix compute budget,
vary model parameters & training tokens



We study:

- **Compute-optimal model sizes**
- **Context length dependence of the compute-optimal model size**

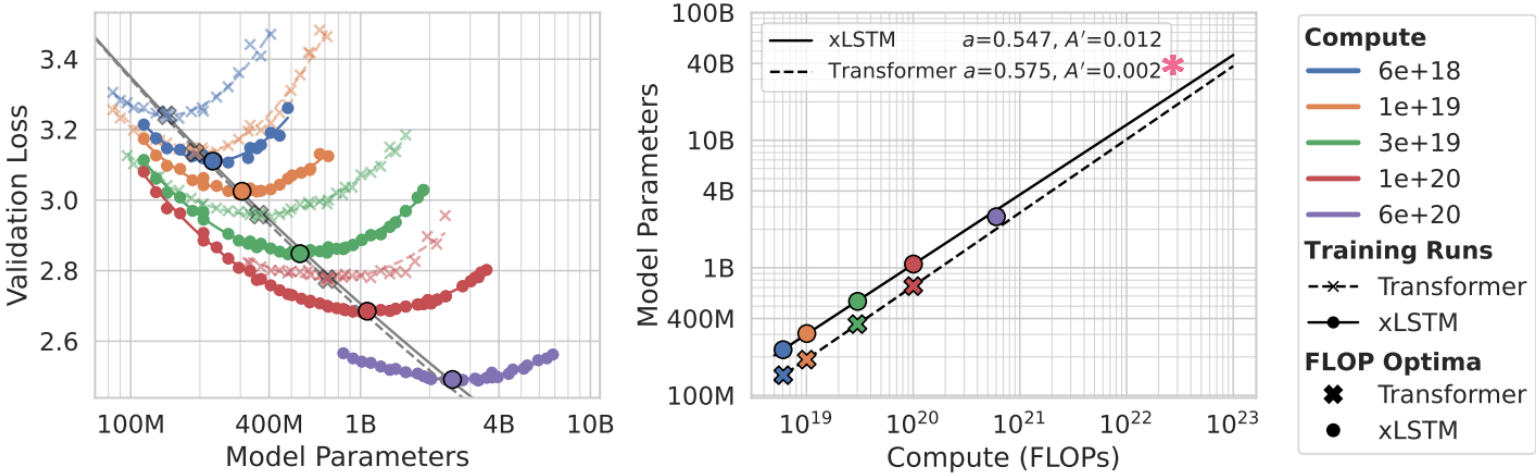
Inference speed

We compare **prefill and generation speed**

[IsoFLOP] Compute-optimal model sizes & context length dependence

- We use the isoFLOP runs with different context length
- Fit a 2nd order polynomial for each compute budget & context length to determine the minimum
- Fit power-laws for parameters / tokens $\hat{N}^*(H) = A' \cdot H^a$ and $\hat{D}^*(H) = B' \cdot H^b$

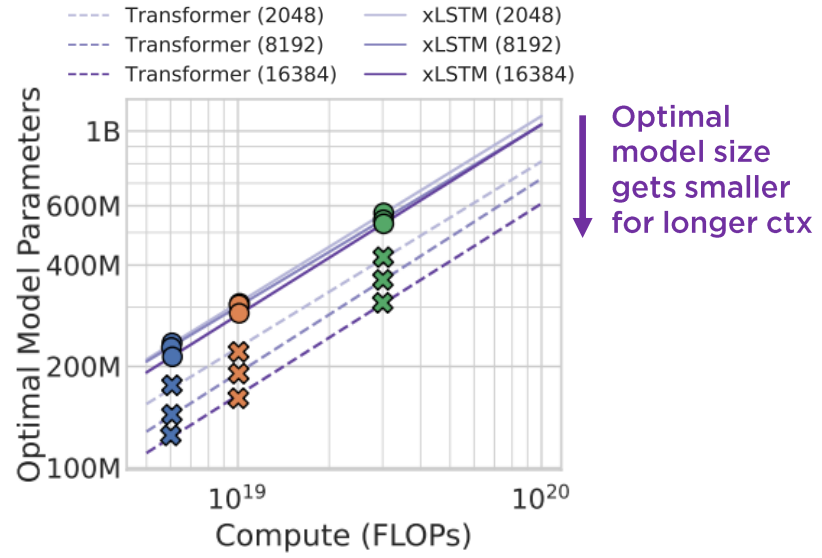
8k context length



➡ xLSTM attains lower loss for the same compute budget

➡ Compute-optimal xLSTM model size is larger

2k, 8k, 16k context lengths



➡ xLSTM can allocate more FLOPs in larger models & training datasets

* Sanity check: Our setup reproduces the power law exponent a for Transformers in [1]

[1] Porian, Wortsman, Jitsev, Schmidt, Carmon (2024) Resolving Discrepancies in Compute-Optimal Scaling of Language Models, NeurIPS

Outline of our scaling law study

Token/param

Fix model parameters,
vary training tokens

IsoFLOP

Fix compute budget,
vary model parameters & training tokens

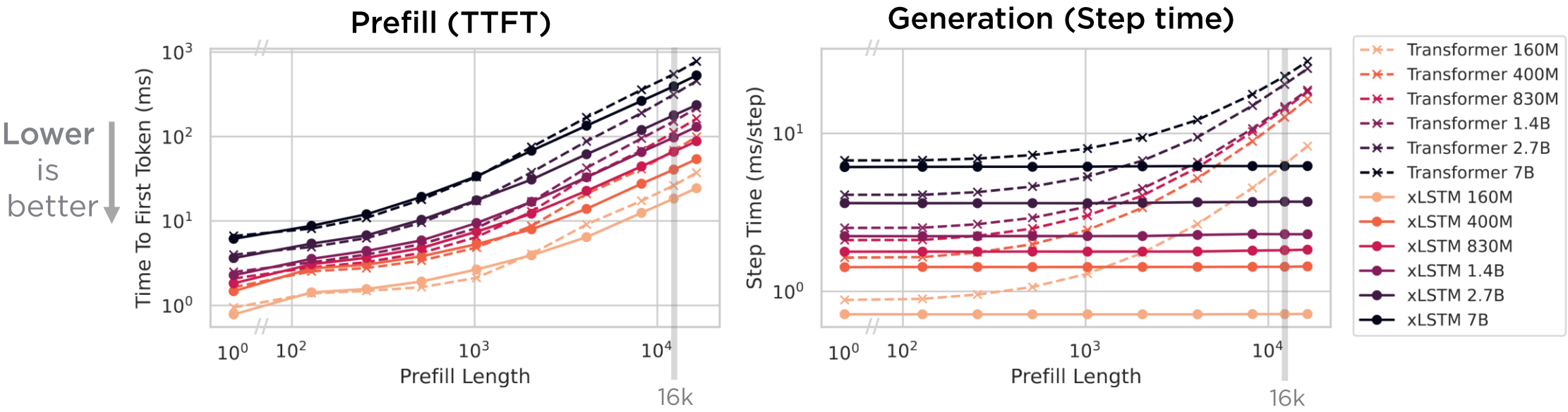
Inference speed

We compare:

- **Prefill speed** in time to first token (TTFT)
- **Generation speed** in step time or tokens/s

Inference speed across model sizes

- We measure inference speeds for different model sizes with batch size 1
  - CUDA Graph optimized Huggingface implementations

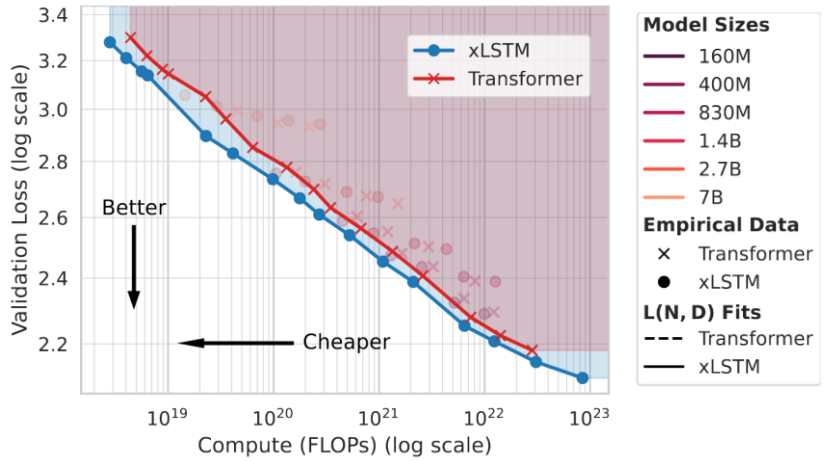


➡ For 16k prefill length,
xLSTM is 30-50% faster than Transformer

➡ For 16k prefill length,
7B xLSTM faster than 160M Transformer

Come and visit us at our poster for more details!

- Exact FLOP, parameter and memory operation counts
- Additional scaling results
- Inference runtime model fits



xLSTM Scaling Laws: Competitive Performance with Linear Time-Complexity

Maximilian Beck [1,2] **Kajetan Schweighofer** [1]
Sebastian Böck [2] **Sebastian Lehner** [1] **Sepp Hochreiter** [1,2]

[1] ELLIS Unit Linz, Institute for Machine Learning, JKU Linz, Austria
[2] NXAI GmbH, Linz, Austria
{beck,schweighofer,slehner}@ml.jku.at

arxiv.org/abs/2510.02228

github.com/NX-AI/xlstm_scaling_laws

JKU | NXAI