xLSTM: Extended Long Short-Term Memory

NeurIPS2024 Spotlight

Maximilian Beck*, beck@ml.jku.at, maxmbeck, maxbeck.ai

Korbinian Pöppel*, poeppel@ml.jku.at, KorbiPoeppel, korbi.ai

Joint work with Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter and Sepp Hochreiter

Vancouver, December 2024

*equal contribution

JOHANNES KEPLER UNIVERSITÄT LINZ

How far do we get in scaling LSTMs to billions of parameters?

Answer: Not so far with the original LSTM!

Why?

The original LSTM has three main limitations:

- L1: Inability to revise storage decisions
- L2: Limited storage capacity
- L3: Lack of parallelizability

L1: Inability to revise storage decisions

Problem:

- Consider a sequence in which you search for an element that is closest to your query element
- As soon as there is a closer element we need to “overwrite” the memory
- The original LSTM struggles with those tasks

Intuition:

- LSTM input gate is bounded due to sigmoid activation and can only change the memory through forget gates over time

L1: Inability to revise storage decisions

Solution:

- We replace the **sigmoid gating** of the original LSTM by **Exponential Gating**

$$c_t = \sigma(\tilde{\mathrm{f}}_t) \; c_{t-1} + \sigma(\tilde{\mathrm{i}}_t) \; \tanh(\tilde{z}_t)$$

$$\Downarrow$$

$$c_t = \sigma(\tilde{\mathrm{f}}_t) \; c_{t-1} + \exp(\tilde{\mathrm{i}}_t) \; \tanh(\tilde{z}_t)$$

JKU | NXAI

L2: Limited Storage Capacity

Problem:

- The storage capacity of the original LSTM is limited, due to the vector memory cell state
- E.g. compare to “infinite” sized KV-cache of Transformers

L2: Limited Storage Capacity

Solution:

- We enhance the vector cell state to a **matrix memory** cell state **with outer product update rule**

$$c_t = \sigma(\tilde{\mathrm{f}}_t)\; c_{t-1} + \sigma(\tilde{\mathrm{i}}_t)\; \tanh(\tilde{z}_t)$$

$$\boldsymbol{C}_t = \sigma(\tilde{\mathrm{f}}_t)\; \boldsymbol{C}_{t-1} + \exp(\tilde{\mathrm{i}}_t)\; \boldsymbol{v}_t\, \boldsymbol{k}_t^\top$$

L3: Lack of Parallelizability

Problem:

- The original LSTM has recurrent weights that connects the hidden state with gate pre-activations
- We need to compute a matrix multiply in every timestep which prohibits parallelizability

L3: Lack of Parallelizability

Solution:

- We drop the recurrent weights and introduce a fully parallelizable variant

xLSTM has two new Memory Cells with Exponential Gating

Exponential Gating

sLSTM

“scalar” cell state

- L1: Exponential Gating

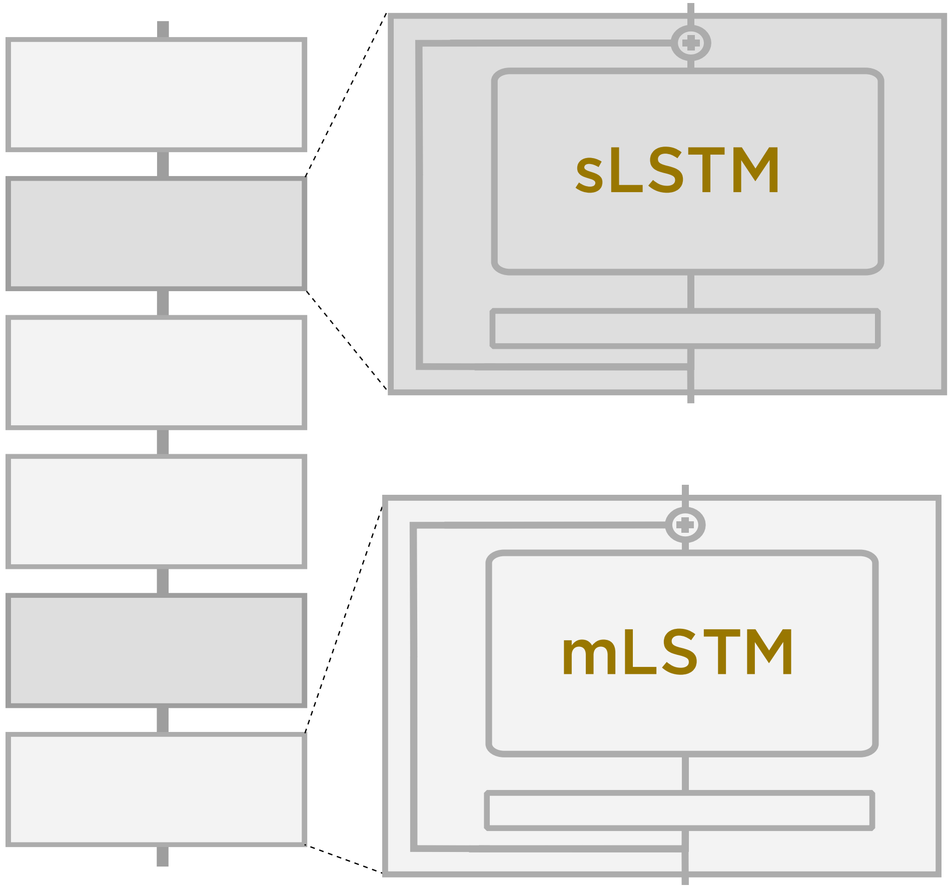
mLSTM

“matrix” cell state

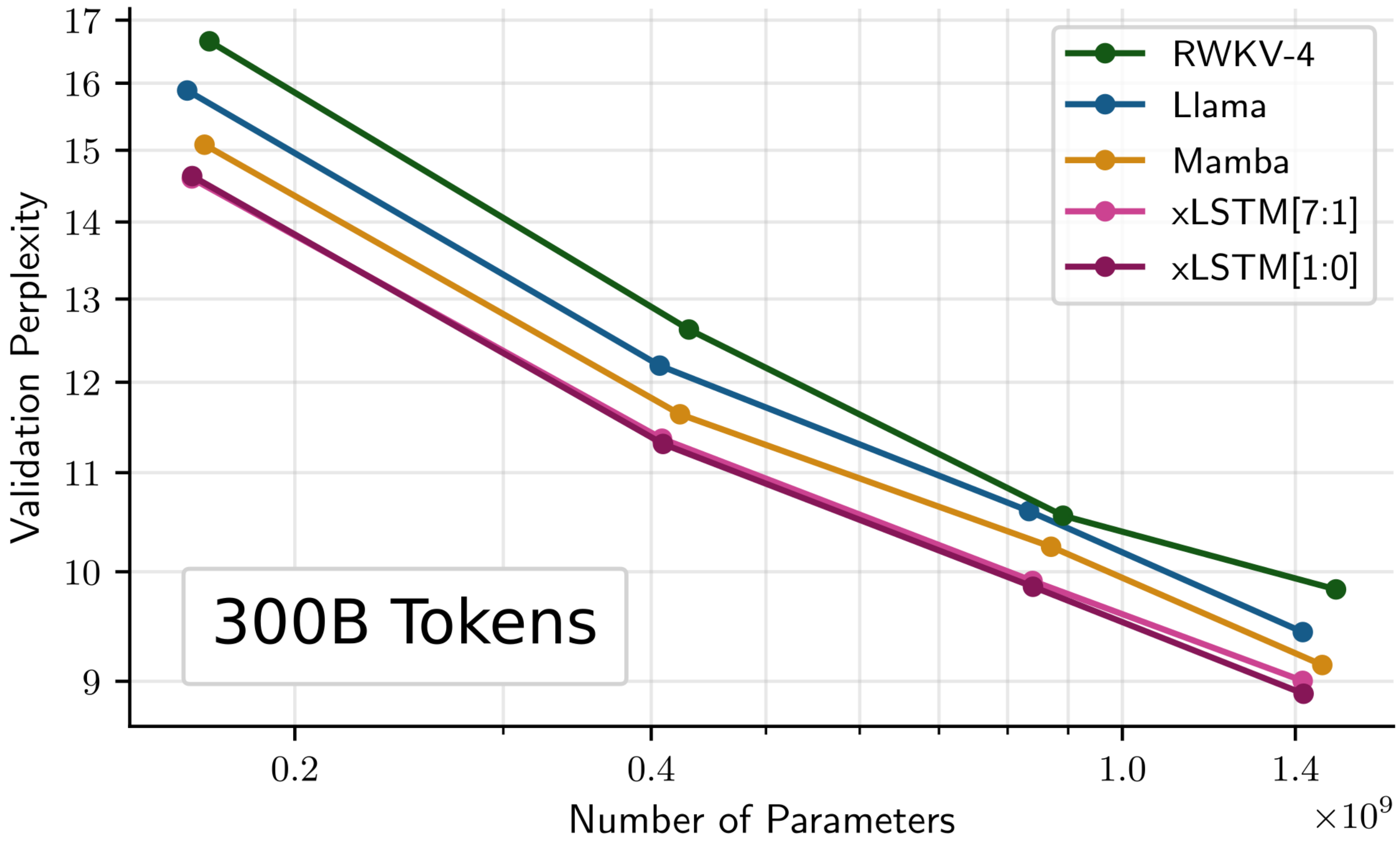
- L1: Exponential Gating
- L2: Matrix Memory
- L3: Parallel Training

xLSTM Architecture

- We use Transformer pre-norm blocks
- We stack mLSTM and sLSTM blocks at a certain ratio



Scaling Behaviour



Let us revisit our initial question:

How far do we get in scaling LSTMs to billions of parameters?

At least as far as current technologies like Transformer or State Space Models.

More Experiments in the Paper...

- A comparison to other recent language modeling architectures
- Experiments on generation times and maximal throughput
- Experiments on length extrapolation
- Evaluations of the xLSTM on language downstream tasks
- Evaluations on synthetic tasks that measure the expressivity of the xLSTM
- Evaluations on synthetic tasks that measure the memory capacity of the xLSTM
- Many more details and formulas...

Thanks for watching!

arxiv.org/abs/2405.04517

github.com/NX-AI/xlstm

Maximilian Beck

beck@ml.jku.at

maxmbeck

maxbeck.ai

Korbinian Pöppel

poeppel@ml.jku.at

KorbiPoeppel

korbi.ai