HPS

jonathan.decker@uni-goettingen.de

Jonathan Decker

# xLSTM, The One To Overcome Transformers

## Should I Be Concerned?

# Short History on LSTM

- Protagonist: Sepp Hochreiter
  - German Computer Scientist
  - 1991 Invented LSTM (Long-Short-Term-Memory)
  - 1997 Published LSTM paper
  - 2024 Founded Startup NX AI in Austria
  - 2024 Invented xLSTM (Extended-Long-Short-Term-Memory)

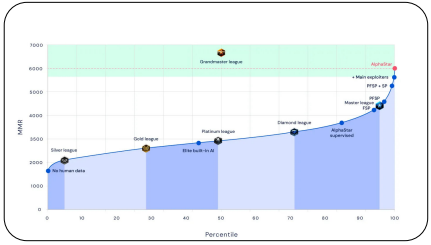
https://www.nx-ai.com/

Hochreiter and Schmidhuber, “Long Short-Term Memory”, 1997
www.netural.com, *European AI Hub*
Beck et al., “xLSTM”, 2024

# LSTM Significance

- OpenAI Five
  - Defeated Dota 2 World Champions
- DeepMind's AlphaStar
  - Reached Grandmaster level (top 0.2%) in StarCraft 2
- Many other applications
  - Machine Translation
  - Time Series Forecasting
  - Sentiment Analysis
  - Text Classification
  - Question-Answering
  - ...

https://openai.com/index/openai-five-defeats-dota-2-world-champions/

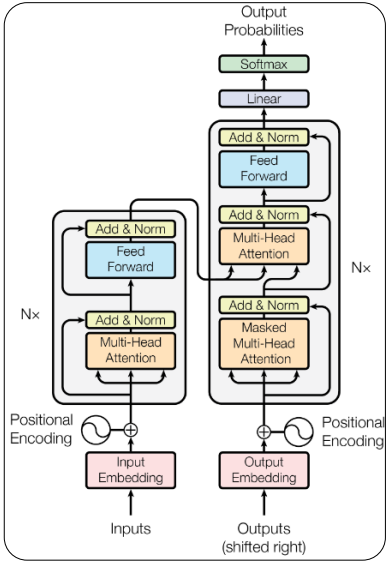



https://www.theverge.com/2019/10/30/20939147/deepmind-google-alphastar-starcraft-2-research-grandma

*OpenAI Five Defeats Dota 2 World Champions*; OpenAI et al., "Dota 2 with Large Scale Deep Reinforcement Learning", 2019
Statt, *DeepMind's StarCraft 2 AI Is Now Better than 99.8 Percent of All Human Players*, 2019; Mathieu et al., "StarCraft II Unplugged: Large Scale Offline Reinforcement Learning"

# Rise of Transformers

- LSTM limitations
  - Limited parallelism
  - Limited storage capacity
  - Inconsistent retrieval
- 2017 Google Transformers arch paper
- 2022 Release of ChatGPT
- 2023-2024 LLM hype
  (most based on Transformers)




*Transformer vs LSTM*, 2023; Vaswani et al., "Attention Is All You Need", 2023

# Appearance of xLSTM

- 2024 xLSTM paper
- Efficiency
  - Transformers has time and memory complexity of $O(N^2)$
  - xLSTM has time complexity $O(N)$ and memory complexity $O(1)$
- Performance
  - Similar or better performance in benchmarks than Transformers!

Beck et al., “xLSTM”, 2024

# Table of contents

1 LSTM Architecture

2 LSTM vs Transformers

3 xLSTM Architecture

4 xLSTM Evaluation

5 xLSTM Resources

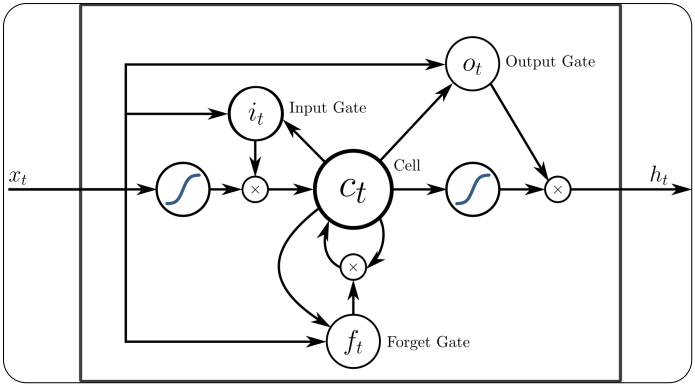
# LSTM Solves Vanishing Gradients Problem

- LSTM is a RNN
  (Recurrent Neural Network)
- VGP (Vanishing Gradient Problem)
  - Occurs when training deep neural networks with gradient-methods and backpropagation
  - For longer sequences, during backpropagation, gradients may become extremely small or large
  - Early layers receive no proper updates
- LSTM *Gated* architecture solves VGP
- Transformers solve VGP via self-attention

*Transformer vs LSTM*, 2023; Hochreiter and Schmidhuber, “Long Short-Term Memory”, 1997

# LSTM Gates

- $i_t$ Input Gate
  - Sigmoid, returns [0-1], storing
- $f_t$ Forget Gate
  - Sigmoid, returns [0-1], forgetting
- $o_t$ Output Gate
  - Sigmoid for what, tanh for scale, forwarding
- $c_t$ Cell State
  - Scalar/Vector storage




https://en.wikipedia.org/wiki/Long_short-term_memory

*Transformer vs LSTM*, 2023; Hochreiter and Schmidhuber, "Long Short-Term Memory", 1997

# Input Processing

- Both Transformers and LSTM
  - Require Tokenization of words, sequences into embedding vectors
- Transformers
  - Consider positional embeddings
  - Process input sequences simultaneously
- LSTM
  - Limited to sequential processing

*Transformer vs LSTM*, 2023; Hochreiter and Schmidhuber, “Long Short-Term Memory”, 1997

# Training

- Transformers
  - Self-attention enables fast, parallel training
  - High memory and compute requirements
- LSTM
  - Only sequential training
  - Reduced memory and compute required

*Transformer vs LSTM*, 2023; Hochreiter and Schmidhuber, “Long Short-Term Memory”, 1997

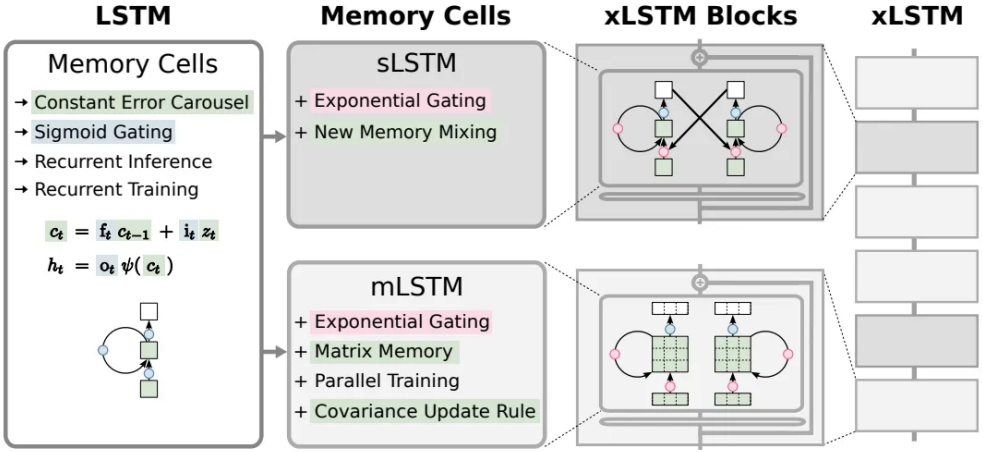



Figure 1: The extended LSTM (xLSTM) family. From left to right: 1. The original LSTM memory cell with constant error carousel and gating. 2. New sLSTM and mLSTM memory cells that introduce exponential gating. sLSTM offers a new memory mixing technique. mLSTM is fully parallelizable with a novel matrix memory cell state and new covariance update rule. 3. mLSTM and sLSTM in residual blocks yield xLSTM blocks. 4. Stacked xLSTM blocks give an xLSTM architecture.

Beck et al., "xLSTM", 2024

# New Mechanisms

- Exponential Gating
  - Replaces Sigmoid
  - Model changes more quickly
  - Includes normalizer state to stabilize exponential gating
- Matrix Memory
  - In mLSTM, replaces scalar memory
  - Enables more storage and parallel processing
  - SLSTM is still sequential
- Flexible Architecture
  - mLSTM and sLSTM blocks can be combined in various ratios

Beck et al., “xLSTM”, 2024; Ahmed, *xLSTM*, 2024

# Efficiency and Performance

- Efficiency
  - Transformers has time and memory complexity of $O(N^2)$
  - xLSTM has time complexity $O(N)$ and memory complexity $O(1)$
- Performance
  - Measured using Perplexity, ability to predict next token, less is better
  - xLSTM outperformed other architectures on a range of benchmarks
  - xLSTM[1:0], only mLSTM and no sLSTM blocks did the best

Beck et al., "xLSTM", 2024; Ahmed, *xLSTM*, 2024

| | Context Sentsitive | | Deterministic Context Free | | Regular | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Bucket Sort | Missing Duplicate | Mod Arithmetic (w Brackets) | Solve Equation | Cycle Nav | Even Pairs | Mod Arithmetic (w/o Brackets) | Parity | Majority | Majority Count |
| Llama | 0.92 ± 0.02 | 0.08 ± 0.0 | 0.02 ± 0.0 | 0.02 ± 0.0 | 0.04 ± 0.01 | 1.0 ± 0.0 | 0.03 ± 0.0 | 0.03 ± 0.01 | 0.37 ± 0.01 | 0.13 ± 0.0 |
| Mamba | 0.69 ± 0.0 | 0.15 ± 0.0 | 0.04 ± 0.01 | 0.05 ± 0.02 | 0.86 ± 0.04 | 1.0 ± 0.0 | 0.05 ± 0.02 | 0.13 ± 0.02 | 0.69 ± 0.01 | 0.45 ± 0.03 |
| Retention | 0.13 ± 0.01 | 0.03 ± 0.0 | 0.03 ± 0.0 | 0.03 ± 0.0 | 0.05 ± 0.01 | 0.51 ± 0.07 | 0.04 ± 0.0 | 0.05 ± 0.01 | 0.36 ± 0.0 | 0.12 ± 0.01 |
| Hyena | 0.3 ± 0.02 | 0.06 ± 0.02 | 0.05 ± 0.0 | 0.02 ± 0.0 | 0.06 ± 0.01 | 0.93 ± 0.07 | 0.04 ± 0.0 | 0.04 ± 0.0 | 0.36 ± 0.01 | 0.18 ± 0.02 |
| RWKV-4 | 0.54 ± 0.0 | 0.21 ± 0.01 | 0.06 ± 0.0 | 0.07 ± 0.0 | 0.13 ± 0.0 | 1.0 ± 0.0 | 0.07 ± 0.0 | 0.06 ± 0.0 | 0.63 ± 0.0 | 0.13 ± 0.0 |
| RWKV-5 | 0.49 ± 0.04 | 0.15 ± 0.01 | 0.08 ± 0.0 | 0.08 ± 0.0 | 0.26 ± 0.05 | 1.0 ± 0.0 | 0.15 ± 0.02 | 0.06 ± 0.03 | 0.73 ± 0.01 | 0.34 ± 0.03 |
| RWKV-6 | 0.96 ± 0.0 | 0.23 ± 0.06 | 0.09 ± 0.01 | 0.09 ± 0.02 | 0.31 ± 0.14 | 1.0 ± 0.0 | 0.16 ± 0.0 | 0.22 ± 0.12 | 0.76 ± 0.01 | 0.24 ± 0.01 |
| LSTM (Block) | 0.99 ± 0.0 | 0.15 ± 0.0 | 0.76 ± 0.0 | 0.5 ± 0.05 | 0.97 ± 0.03 | 1.0 ± 0.0 | 0.91 ± 0.09 | 1.0 ± 0.0 | 0.58 ± 0.02 | 0.27 ± 0.0 |
| LSTM | 0.94 ± 0.01 | 0.2 ± 0.0 | 0.72 ± 0.04 | 0.38 ± 0.05 | 0.93 ± 0.07 | 1.0 ± 0.0 | 1.0 ± 0.0 | 1.0 ± 0.0 | 0.82 ± 0.02 | 0.33 ± 0.0 |
| xLSTM[0:1] | 0.84 ± 0.08 | 0.23 ± 0.01 | 0.57 ± 0.09 | 0.55 ± 0.09 | 1.0 ± 0.0 | 1.0 ± 0.0 | 1.0 ± 0.0 | 1.0 ± 0.0 | 0.75 ± 0.02 | 0.22 ± 0.0 |
| xLSTM[1:0] | 0.97 ± 0.0 | 0.33 ± 0.22 | 0.03 ± 0.0 | 0.03 ± 0.01 | 0.86 ± 0.01 | 1.0 ± 0.0 | 0.04 ± 0.0 | 0.04 ± 0.01 | 0.74 ± 0.01 | 0.46 ± 0.0 |
| xLSTM[1:1] | 0.7 ± 0.21 | 0.2 ± 0.01 | 0.15 ± 0.06 | 0.24 ± 0.04 | 0.8 ± 0.03 | 1.0 ± 0.0 | 0.6 ± 0.4 | 1.0 ± 0.0 | 0.64 ± 0.04 | 0.5 ± 0.0 |

Beck et al., “xLSTM”, 2024

# Limitations

- sLSTM does not support parallelization
  - CUDA implementation provided is 1.5 times slower than mLSTM
- mLSTM CUDA kernels are not optimized, could be 4 times faster
- mLSTM has high computation complexity due to *dxd* matrices
- Matrix mem is independent of context length
  - Might be problem when scaling context length for constant matrix mem
- xLSTM arch and hyperparameters have not been optimized yet, full potential of xLSTM has not been achieved
- Experiments only looked at small LLMs, up to 1.3B parameters
- Benchmarks are not independent
  - Other archs not sufficiently optimized
  - Performance gain LSTM to xLSTM not significant enough

Beck et al., "xLSTM", 2024; Ahmed, *xLSTM*, 2024

# Consqeuences

- Memory and compute efficiency improvement is significant
  - Especially compared to Transformers
  - Could replace Transformers for very small and very large LLMs
- xLSTM is flexible
  - Different combinations of sLSTM and mLSTM might be better at specific tasks
- xLSTM has the potential to challenge the status quo
  - Not only/(mostly) transformer-based LLMs in the future

Beck et al., “xLSTM”, 2024; Ahmed, *xLSTM*, 2024

# xLSTM on HPC

glogin9 using NHR stack

```
git clone https://github.com/NX-AI/xlstm
cd xlstm
module load miniconda3
conda env create -n xlstm -f environment_pt220cu121.yaml
source $CONDASH # Might need this for conda init bash
conda activate xlstm
pip install numpy==1.26.4
module load cuda

srun -p kisski --pty -n 1 -c 64 -t 1:00:00 -G A100:1 bash
export PYTHONPATH=$(pwd)
python experiments/main.py --config experiments/parity_xlstm10.yaml
```

# xLSTM Resources

- Paper: https://arxiv.org/abs/2405.04517
- Resources: https://github.com/AI-Guru/xlstm-resources
- Official code with CUDA: https://github.com/NX-AI/xlstm
- Pure Python code: https://github.com/muditbhargava66/PyxLSTM
- xLSTM (with Memes:) https://www.youtube.com/watch?v=7r1-4LDJDmM
- OpenAI Five Dota2 Finals: https://www.twitch.tv/videos/410533063
- AlphaStar Demo: https://www.youtube.com/watch?v=cUTMhmVh1qs

# References I

Ahmed, Zul. *xLSTM: Enhancing Long Short-Term Memory for Large Language Models*. Medium. May 8, 2024. URL: https://medium.com/@zahmed333/xlstm-enhancing-long-short-term-memory-for-large-language-models-b9f0f07618aa (visited on 07/23/2024).

Beck, Maximilian et al. "xLSTM: Extended Long Short-Term Memory". In: (May 7, 2024). DOI: 10.48550/arXiv.2405.04517. arXiv: 2405.04517 [cs, stat]. URL: http://arxiv.org/abs/2405.04517 (visited on 07/23/2024). Pre-published.

Hochreiter, Sepp and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 15, 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (visited on 07/23/2024).

Mathieu, Michaël et al. "StarCraft II Unplugged: Large Scale Offline Reinforcement Learning". In: ().

OpenAI et al. "Dota 2 with Large Scale Deep Reinforcement Learning". In: (Dec. 13, 2019). DOI: 10.48550/arXiv.1912.06680. arXiv: 1912.06680 [cs, stat]. URL: http://arxiv.org/abs/1912.06680 (visited on 07/23/2024). Pre-published.

*OpenAI Five Defeats Dota 2 World Champions*. URL: https://openai.com/index/openai-five-defeats-dota-2-world-champions/ (visited on 07/23/2024).

# References II

Statt, Nick. *DeepMind's StarCraft 2 AI Is Now Better than 99.8 Percent of All Human Players*. The Verge. Oct. 30, 2019. URL: https://www.theverge.com/2019/10/30/20939147/deepmind-google-alphastar-starcraft-2-research-grandmaster-level (visited on 07/23/2024).

*Transformer vs LSTM: A Helpful Illustrated Guide – Be on the Right Side of Change*. July 4, 2023. URL: https://blog.finxter.com/transformer-vs-lstm/ (visited on 07/23/2024).

Vaswani, Ashish et al. "Attention Is All You Need". In: (Aug. 1, 2023). arXiv: 1706.03762 [cs]. URL: http://arxiv.org/abs/1706.03762 (visited on 07/23/2024). Pre-published.

www.netural.com, Netural-. *European AI Hub: Leading Research for AI Technologies and Solutions*. URL: https://www.nx-ai.com (visited on 07/23/2024).

# LSTM

$$c_t = \mathrm{f}_t \; c_{t-1} + \mathrm{i}_t \; z_t \qquad \text{cell state} \qquad (2)$$

$$h_t = \mathrm{o}_t \; \tilde{h}_t \,, \qquad \tilde{h}_t = \psi\left(c_t\right) \qquad \text{hidden state} \qquad (3)$$

$$z_t = \varphi\left(\tilde{z}_t\right) \,, \qquad \tilde{z}_t = \boldsymbol{w}_z^\top \boldsymbol{x}_t + r_z h_{t-1} + b_z \qquad \text{cell input} \qquad (4)$$

$$\mathrm{i}_t = \sigma\left(\tilde{\mathrm{i}}_t\right) \,, \qquad \tilde{\mathrm{i}}_t = \boldsymbol{w}_\mathrm{i}^\top \boldsymbol{x}_t + r_\mathrm{i} \, h_{t-1} + b_\mathrm{i} \qquad \text{input gate} \qquad (5)$$

$$\mathrm{f}_t = \sigma\left(\tilde{\mathrm{f}}_t\right) \,, \qquad \tilde{\mathrm{f}}_t = \boldsymbol{w}_\mathrm{f}^\top \boldsymbol{x}_t + r_\mathrm{f} \, h_{t-1} + b_\mathrm{f} \qquad \text{forget gate} \qquad (6)$$

$$\mathrm{o}_t = \sigma\left(\tilde{\mathrm{o}}_t\right) \,, \qquad \tilde{\mathrm{o}}_t = \boldsymbol{w}_\mathrm{o}^\top \boldsymbol{x}_t + r_\mathrm{o} \, h_{t-1} + b_\mathrm{o} \qquad \text{output gate} \qquad (7)$$

# sLSTM

$$c_t = \mathrm{f}_t \; c_{t-1} + \mathrm{i}_t \; z_t \qquad \text{cell state} \qquad (8)$$

$$n_t = \mathrm{f}_t \; n_{t-1} + \mathrm{i}_t \qquad \text{normalizer state} \qquad (9)$$

$$h_t = \mathrm{o}_t \; \tilde{h}_t \,, \qquad \tilde{h}_t = c_t \,/\, n_t \qquad \text{hidden state} \qquad (10)$$

$$z_t = \varphi\left(\tilde{z}_t\right) \,, \qquad \tilde{z}_t = \boldsymbol{w}_z^\top \boldsymbol{x}_t + r_z h_{t-1} + b_z \qquad \text{cell input} \qquad (11)$$

$$\mathrm{i}_t = \exp\left(\tilde{\mathrm{i}}_t\right) \,, \qquad \tilde{\mathrm{i}}_t = \boldsymbol{w}_\mathrm{i}^\top \boldsymbol{x}_t + r_\mathrm{i} \, h_{t-1} + b_\mathrm{i} \qquad \text{input gate} \qquad (12)$$

$$\mathrm{f}_t = \sigma\left(\tilde{\mathrm{f}}_t\right) \text{ OR } \exp\left(\tilde{\mathrm{f}}_t\right) \,, \qquad \tilde{\mathrm{f}}_t = \boldsymbol{w}_\mathrm{f}^\top \boldsymbol{x}_t + r_\mathrm{f} \, h_{t-1} + b_\mathrm{f} \qquad \text{forget gate} \qquad (13)$$

$$\mathrm{o}_t = \sigma\left(\tilde{\mathrm{o}}_t\right) \,, \qquad \tilde{\mathrm{o}}_t = \boldsymbol{w}_\mathrm{o}^\top \boldsymbol{x}_t + r_\mathrm{o} \, h_{t-1} + b_\mathrm{o} \qquad \text{output gate} \qquad (14)$$

$$m_t = \max\left(\log(\mathrm{f}_t) + m_{t-1}, \log(\mathrm{i}_t)\right) \qquad \text{stabilizer state} \qquad (15)$$

$$\mathrm{i}'_t = \exp\left(\log\left(\mathrm{i}_t\right) - m_t\right) = \exp\left(\tilde{\mathrm{i}}_t - m_t\right) \qquad \text{stabil. input gate} \qquad (16)$$

$$\mathrm{f}'_t = \exp\left(\log\left(\mathrm{f}_t\right) + m_{t-1} - m_t\right) \qquad \text{stabil. forget gate} \qquad (17)$$

# mLSTM

$$\boldsymbol{C}_t = \mathrm{f}_t \, \boldsymbol{C}_{t-1} + \mathrm{i}_t \, \boldsymbol{v}_t \, \boldsymbol{k}_t^\top \qquad \text{cell state} \quad (19)$$

$$\boldsymbol{n}_t = \mathrm{f}_t \, \boldsymbol{n}_{t-1} + \mathrm{i}_t \, \boldsymbol{k}_t \qquad \text{normalizer state} \quad (20)$$

$$\boldsymbol{h}_t = \mathbf{o}_t \odot \tilde{\boldsymbol{h}}_t \,, \qquad \tilde{\boldsymbol{h}}_t = \boldsymbol{C}_t \, \boldsymbol{q}_t \; / \; \max\left\{ \left| \boldsymbol{n}_t^\top \, \boldsymbol{q}_t \right|, 1 \right\} \qquad \text{hidden state} \quad (21)$$

$$\boldsymbol{q}_t = \boldsymbol{W}_q \, \boldsymbol{x}_t + \boldsymbol{b}_q \qquad \text{query input} \quad (22)$$

$$\boldsymbol{k}_t = \frac{1}{\sqrt{d}} \boldsymbol{W}_k \, \boldsymbol{x}_t + \boldsymbol{b}_k \qquad \text{key input} \quad (23)$$

$$\boldsymbol{v}_t = \boldsymbol{W}_v \, \boldsymbol{x}_t + \boldsymbol{b}_v \qquad \text{value input} \quad (24)$$

$$\mathrm{i}_t = \exp\left(\tilde{\mathrm{i}}_t\right) \,, \qquad \tilde{\mathrm{i}}_t = \boldsymbol{w}_\mathrm{i}^\top \, \boldsymbol{x}_t + b_\mathrm{i} \qquad \text{input gate} \quad (25)$$

$$\mathrm{f}_t = \sigma\left(\tilde{\mathrm{f}}_t\right) \text{ OR } \exp\left(\tilde{\mathrm{f}}_t\right), \quad \tilde{\mathrm{f}}_t = \boldsymbol{w}_\mathrm{f}^\top \, \boldsymbol{x}_t + b_\mathrm{f} \qquad \text{forget gate} \quad (26)$$

$$\mathbf{o}_t = \sigma\left(\tilde{\mathbf{o}}_t\right) \,, \qquad \tilde{\mathbf{o}}_t = \boldsymbol{W}_\mathbf{o} \, \boldsymbol{x}_t + \boldsymbol{b}_\mathbf{o} \qquad \text{output gate} \quad (27)$$