# Transformers are RNNs:
# Fast Autoregressive Transformers with Linear Attention

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas and François Fleuret

ICML, July 2020

https://linear-transformers.com/

Funded by FNSNF

# Transformers are performant

Transformer models have demonstrated impressive performance on

- NLP (Vaswani et al., 2017; Devlin et al., 2019; Dai et al., 2019; Yang et al., 2019; Radford et al., 2019)
  - Neural Machine Translation
  - Question Answering
  - Textual Entailment

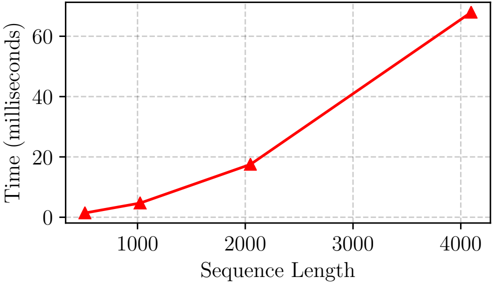
# Transformers are performant

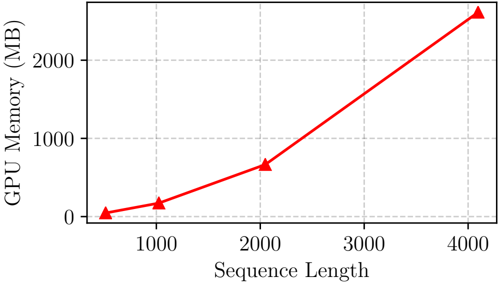
Transformer models have demonstrated impressive performance on

- NLP (Vaswani et al., 2017; Devlin et al., 2019; Dai et al., 2019; Yang et al., 2019; Radford et al., 2019)
  - Neural Machine Translation
  - Question Answering
  - Textual Entailment
- Speech & audio processing (Sperber et al., 2018)
- Autoregressive image generation and general computer vision (Child et al., 2019; Parmar et al., 2019; Carion et al., 2020; Cordonnier et al., 2020)

# Transformers are hard to scale

Self-attention **computation and memory scales** as $\mathcal{O}\left(N^2\right)$ with respect to the **sequence length**.




A single self-attention layer in an NVIDIA GTX 1080 Ti

# Our contributions in a nutshell

- A transformer model with **linear complexity** both for memory and computation **during training**

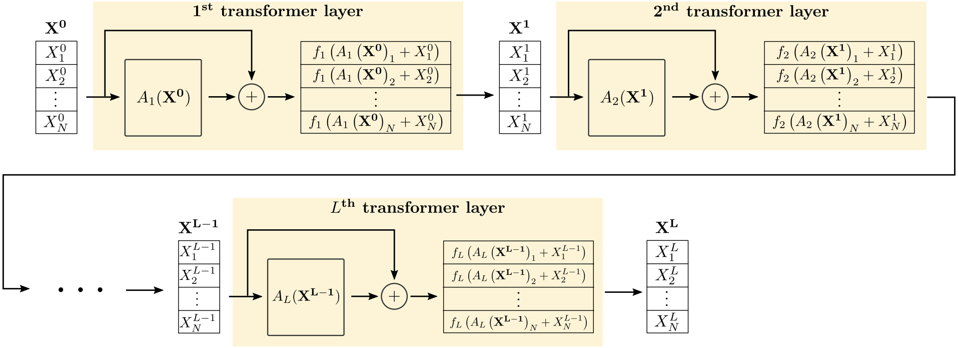
# Our contributions in a nutshell

- A transformer model with **linear complexity** both for memory and computation **during training**
- A transformer model with **linear computational complexity and constant memory** for **autoregressive inference**

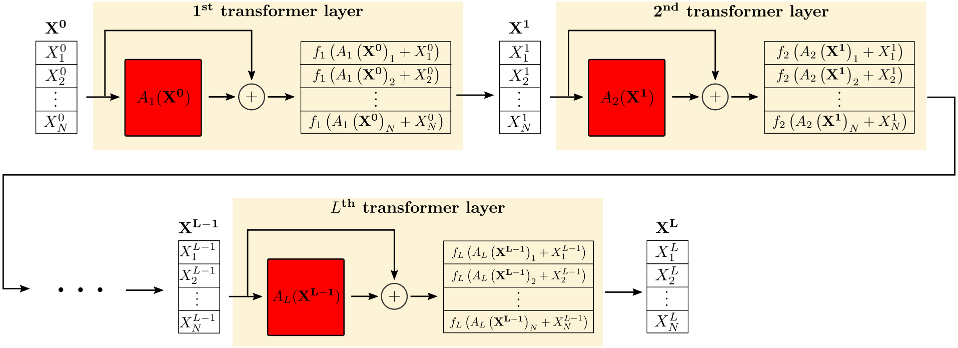
# Our contributions in a nutshell

- A transformer model with **linear complexity** both for memory and computation **during training**
- A transformer model with **linear computational complexity and constant memory** for **autoregressive inference**
- Unravel the **relation between transformers and RNNs**

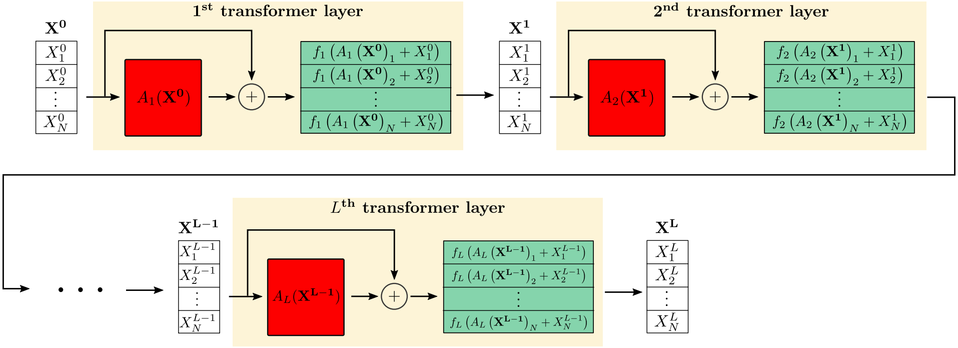
# Definition of a transformer



# Definition of a transformer



# Definition of a transformer



# Self-Attention

The commonly used attention mechanism is the scaled dot product attention

$$Q = XW_Q$$
$$K = XW_K$$
$$V = XW_V$$
$$A_l(X) = V' = \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right)V$$

# Self-Attention

The commonly used attention mechanism is the scaled dot product attention

$$
\begin{aligned}
Q &= XW_Q \\
K &= XW_K \\
V &= XW_V \\
A_l(X) = V' &= \operatorname{softmax}\left(\frac{QK^T}{\sqrt{D}}\right)V
\end{aligned}
$$

# Self-Attention

The commonly used attention mechanism is the scaled dot product attention

$$Q = XW_Q$$
$$K = XW_K$$
$$V = XW_V$$
$$A_l(X) = V' = \operatorname{softmax}\left(\boxed{\frac{QK^T}{\sqrt{D}}}\right)V$$

Quadratic complexity

# Linear Attention

What if we write the self-attention using an **arbitrary similarity score?**

$$V_i' = \frac{\sum_{j=1}^{N} \mathrm{sim}\left(Q_i, K_j\right) V_j}{\sum_{j=1}^{N} \mathrm{sim}\left(Q_i, K_j\right)}$$

# Linear Attention

What if this similarity is a kernel, namely $\mathrm{sim}(a, b) = \phi(a)^T \phi(b)$?

$$
\begin{aligned}
V_i' &= \frac{\sum_{j=1}^{N} \mathrm{sim}(Q_i, K_j)\, V_j}{\sum_{j=1}^{N} \mathrm{sim}(Q_i, K_j)} \\
&= \frac{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)\, V_j}{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)}
\end{aligned}
$$

Kernelization

# Linear Attention

**Matrix products are associative** which makes the attention computation $\mathcal{O}(N)$ with respect to the sequence length.

$$\begin{aligned}
V_i' &= \frac{\sum_{j=1}^{N} \mathrm{sim}\left(Q_i, K_j\right) V_j}{\sum_{j=1}^{N} \mathrm{sim}\left(Q_i, K_j\right)} \\
&\qquad\qquad\qquad\qquad \text{Kernelization} \\
&= \frac{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right) V_j}{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right)} \\
&\qquad\qquad\qquad\qquad \text{Associativity property} \\
&= \frac{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right) V_j^T}{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right)}
\end{aligned}$$

# Causal Masking

Causal masking is used to efficiently train autoregressive transformers.

# Causal Masking

Causal masking is used to efficiently train autoregressive transformers.

**Non-autoregressive**

$$V_i' = \frac{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right) V_j}{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right)}$$

**Autoregressive**

$$V_i' = \frac{\sum_{j=1}^{i} \text{sim}\left(Q_i, K_j\right) V_j}{\sum_{j=1}^{i} \text{sim}\left(Q_i, K_j\right)}$$

# Causal Masking

Causal masking is used to efficiently train autoregressive transformers.

**Non-autoregressive**

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)}$$

**Autoregressive**

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j)}$$

# Causal Masking

Causal masking is used to efficiently train autoregressive transformers.

**Non-autoregressive**

$$V_i' = \frac{\phi(Q_i)^T \overbrace{\sum_{j=1}^{N} \phi(K_j) V_j^T}^{S}}{\phi(Q_i)^T \underbrace{\sum_{j=1}^{N} \phi(K_j)}_{Z}}$$

**Autoregressive**

$$V_i' = \frac{\phi(Q_i)^T \overbrace{\sum_{j=1}^{i} \phi(K_j) V_j^T}^{S_i}}{\phi(Q_i)^T \underbrace{\sum_{j=1}^{i} \phi(K_j)}_{Z_i}}$$

# Causal Masking

Causal masking is used to efficiently train autoregressive transformers.

**Non-autoregressive**

$$V_i' = \frac{\phi\left(Q_i\right)^T \overbrace{\sum_{j=1}^{N} \phi\left(K_j\right) V_j^T}^{S}}{\phi\left(Q_i\right)^T \underbrace{\sum_{j=1}^{N} \phi\left(K_j\right)}_{Z}}$$

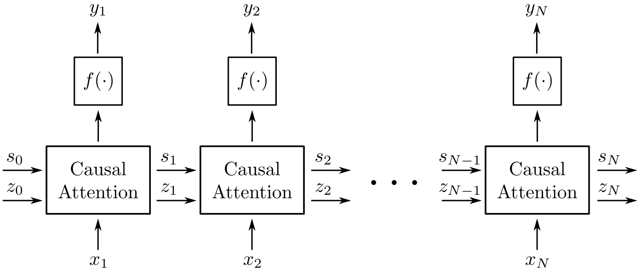
**Autoregressive**

$$V_i' = \frac{\phi\left(Q_i\right)^T \overbrace{\sum_{j=1}^{i} \phi\left(K_j\right) V_j^T}^{S_i}}{\phi\left(Q_i\right)^T \underbrace{\sum_{j=1}^{i} \phi\left(K_j\right)}_{Z_i}}$$

Naive computation of $S_i$ and $Z_i$ results in quadratic complexity.

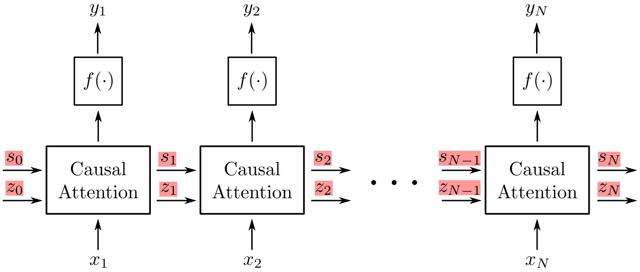
# Transformers are RNNs

Autoregressive transformers can be written as a function that **receives an input** $x_i$, **modifies the internal state** $\{s_{i-1}, z_{i-1}\}$ and **predicts an output** $y_i$.



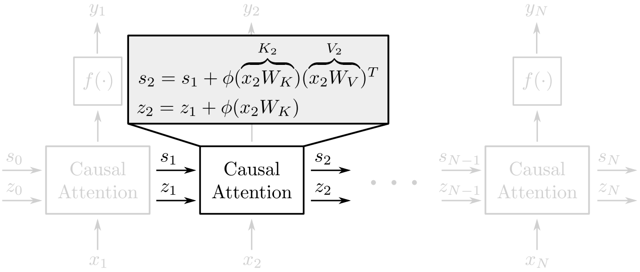
# Transformers are RNNs

Autoregressive transformers can be written as a function that **receives an input** $x_i$, **modifies the internal state** $\{s_{i-1}, z_{i-1}\}$ and **predicts an output** $y_i$.



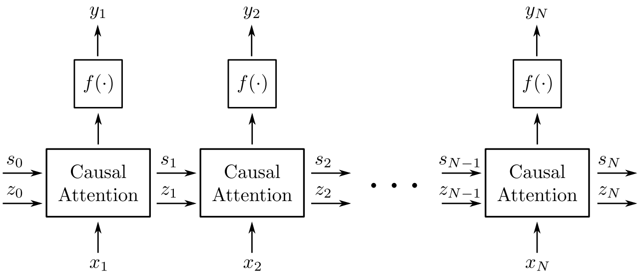
# Transformers are RNNs

Autoregressive transformers can be written as a function that **receives an input** $x_i$, **modifies the internal state** $\{s_{i-1}, z_{i-1}\}$ and **predicts an output** $y_i$.



# Transformers are RNNs

Autoregressive transformers can be written as a function that **receives an input** $x_i$, **modifies the internal state** $\{s_{i-1}, z_{i-1}\}$ and **predicts an output** $y_i$.




Autoregressive inference with **linear complexity and constant memory**.

# Practical implications

- Our **theoretical analysis holds for all transformers** even when using infinite dimensional feature maps

# Practical implications

- Our **theoretical analysis holds for all transformers** even when using infinite dimensional feature maps
- We need a simple **finite dimensional feature map** to speed up computation

# Practical implications

- Our **theoretical analysis holds for all transformers** even when using infinite dimensional feature maps
- We need a simple **finite dimensional feature map** to speed up computation
- We **derive the gradients as cumulative sums** which allows for a significant speed-up

# Experimental setup

Baselines

- Softmax transformer (Vaswani et al., 2017)
- LSH attention from Reformer (Kitaev et al., 2020)

Experiments

- Artificial benchmark for computational and memory requirements
- Autoregressive image generation on MNIST and CIFAR-10
- Automatic speech recognition on Wall Street Journal

# Experimental setup

Baselines

- Softmax transformer (Vaswani et al., 2017)
- LSH attention from Reformer (Kitaev et al., 2020)

Experiments

- Artificial benchmark for computational and memory requirements
- Autoregressive image generation on MNIST and CIFAR-10
- Automatic speech recognition on Wall Street Journal

# Experimental setup

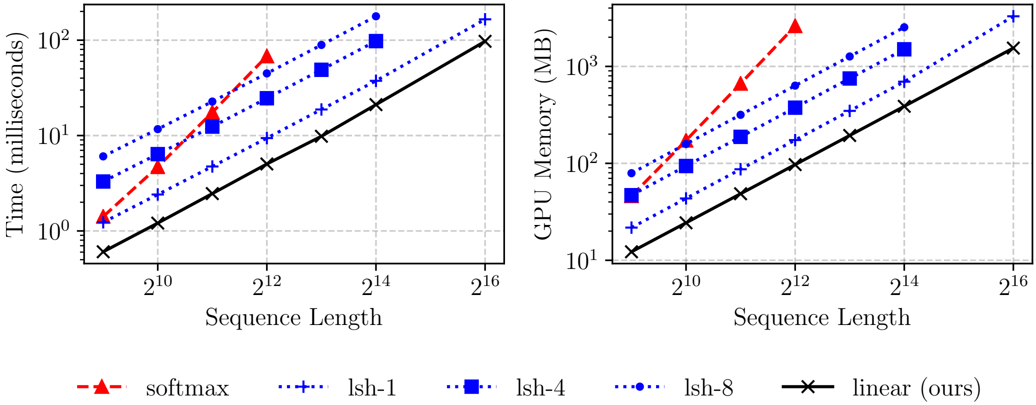
Baselines

- Softmax transformer (Vaswani et al., 2017)
- LSH attention from Reformer (Kitaev et al., 2020)

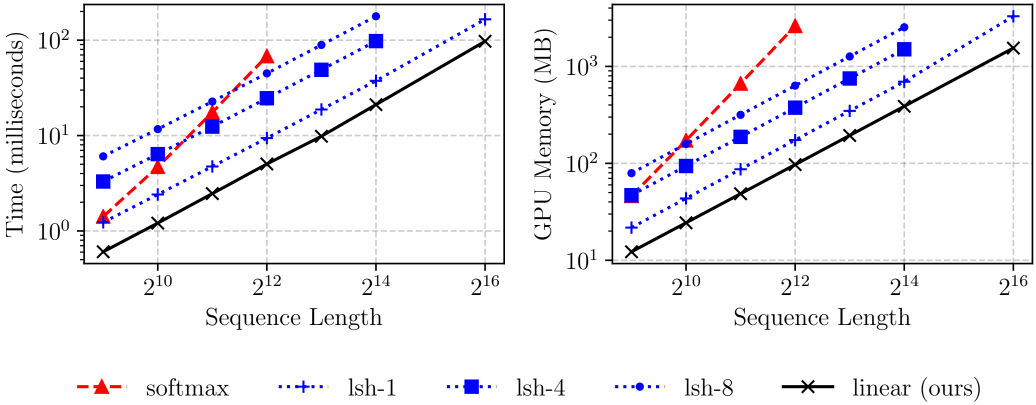
Experiments

- Artificial benchmark for computational and memory requirements
- Autoregressive image generation on MNIST and CIFAR-10
- Automatic speech recognition on Wall Street Journal

# Benchmark


Time (milliseconds)
$10^0$
$10^1$
$10^2$
$2^{10}$
$2^{12}$
$2^{14}$
$2^{16}$
Sequence Length
GPU Memory (MB)
$10^1$
$10^2$
$10^3$
$2^{10}$
$2^{12}$
$2^{14}$
$2^{16}$
Sequence Length
softmax
lsh-1
lsh-4
lsh-8
linear (ours)

# Benchmark



Time (milliseconds)
$10^0$
$10^1$
$10^2$
$2^{10}$
$2^{12}$
$2^{14}$
$2^{16}$
Sequence Length
GPU Memory (MB)
$10^1$
$10^2$
$10^3$
$2^{10}$
$2^{12}$
$2^{14}$
$2^{16}$
Sequence Length
softmax
lsh-1
lsh-4
lsh-8
linear (ours)

# Autoregressive image generation

**Unconditional samples after 250 epochs on MNIST**

Ours (0.644 bpd)

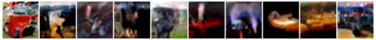
Softmax (0.621 bpd)

LSH-1 (0.745 bpd)

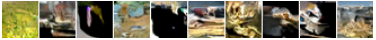
LSH-4 (0.676 bpd)

**Unconditional samples after 1 GPU week on CIFAR-10**

Ours (3.40 bpd)

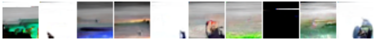


Softmax (3.47 bpd)



LSH-1 (3.39 bpd)

LSH-4 (3.51 bpd)



# Autoregressive image generation

**Unconditional samples after 250 epochs on MNIST**

Ours (0.644 bpd)

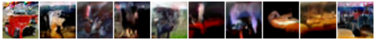
Softmax (0.621 bpd)

LSH-1 (0.745 bpd)

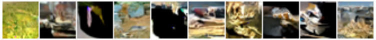
LSH-4 (0.676 bpd)

**Unconditional samples after 1 GPU week on CIFAR-10**

Ours (3.40 bpd)

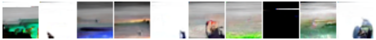


Softmax (3.47 bpd)



LSH-1 (3.39 bpd)

LSH-4 (3.51 bpd)



# Autoregressive image generation

**Unconditional samples after 250 epochs on MNIST**

Ours (0.644 bpd)

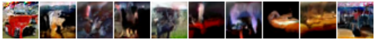
Softmax (0.621 bpd)

LSH-1 (0.745 bpd)

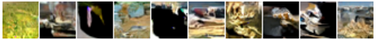
LSH-4 (0.676 bpd)

**Unconditional samples after 1 GPU week on CIFAR-10**

Ours (3.40 bpd)

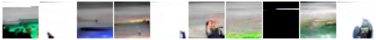


Softmax (3.47 bpd)

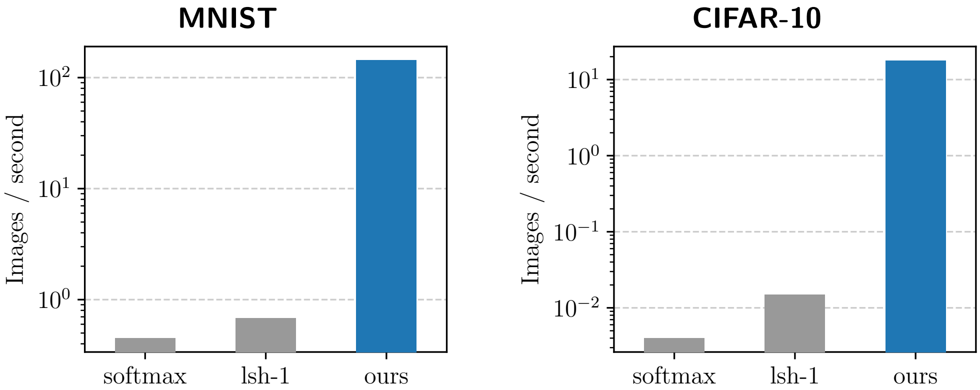


LSH-1 (3.39 bpd)

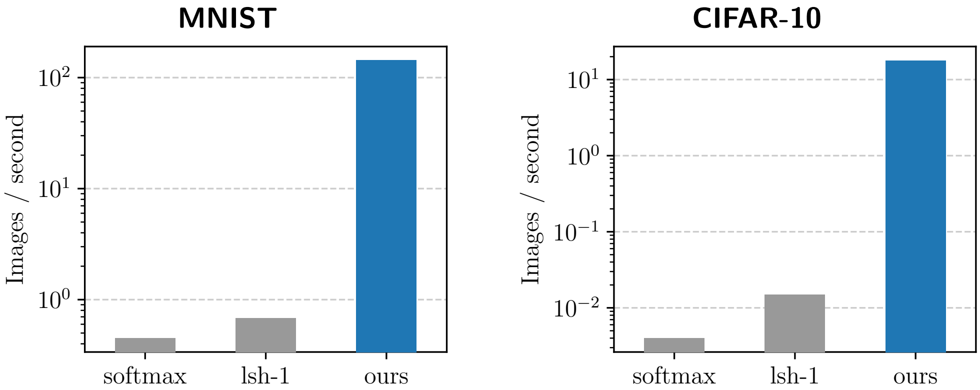
LSH-4 (3.51 bpd)



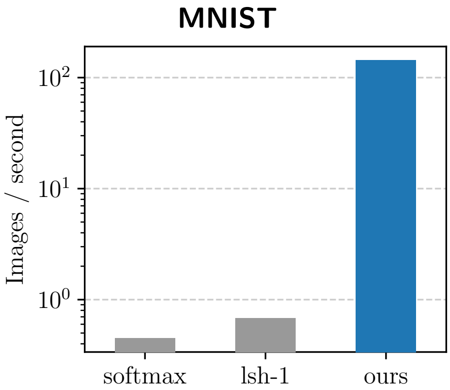
# Autoregressive image generation

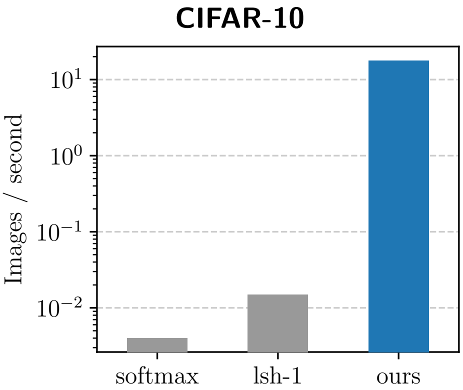


# Autoregressive image generation

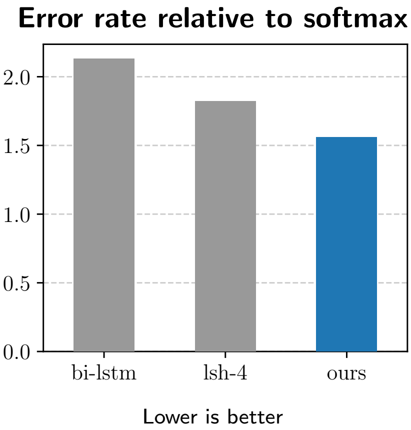


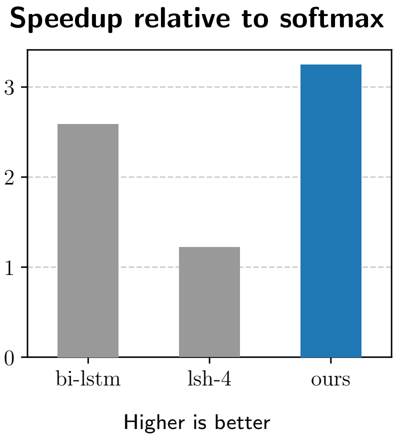
# Autoregressive image generation



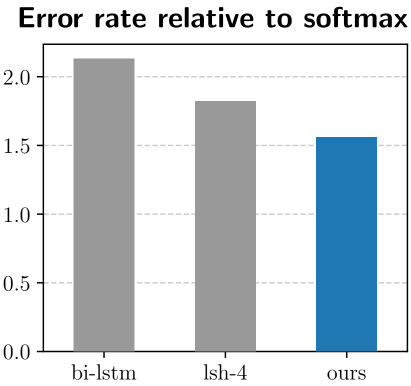



# Automatic speech recognition

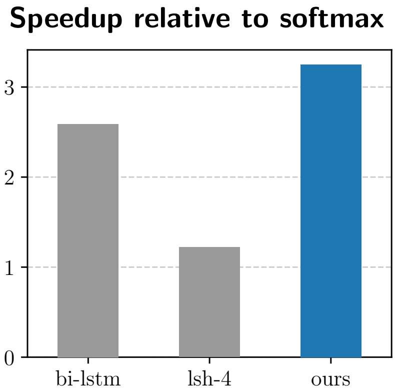


# Automatic speech recognition

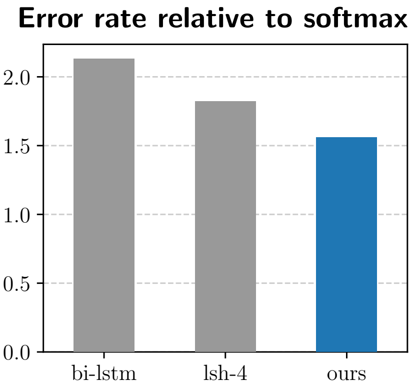



Lower is better

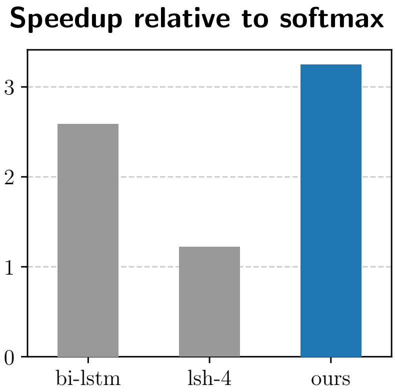



Higher is better

# Automatic speech recognition

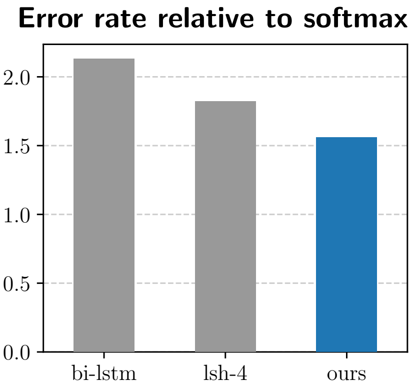



Lower is better

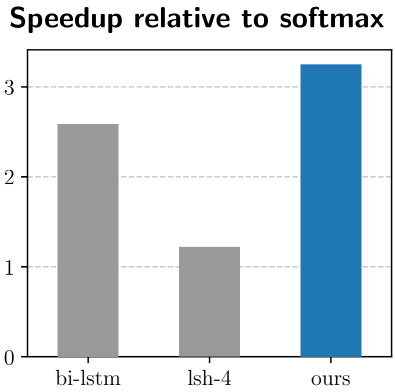



Higher is better

# Automatic speech recognition




Lower is better




Higher is better

# Summary

- **Kernel feature maps** and **matrix associativity** yield an attention with linear complexity.
- Computing the key value matrix as a **cumulative sum** extends our efficient attention computation to the autoregressive case
- Using the RNN formulation to perform autoregressive inference requires **constant memory** and is **many times faster**

# Summary

- **Kernel feature maps** and **matrix associativity** yield an attention with linear complexity.
- Computing the key value matrix as a **cumulative sum** extends our efficient attention computation to the autoregressive case
- Using the RNN formulation to perform autoregressive inference requires **constant memory** and is **many times faster**

# Summary

- **Kernel feature maps** and **matrix associativity** yield an attention with linear complexity.
- Computing the key value matrix as a **cumulative sum** extends our efficient attention computation to the autoregressive case
- Using the RNN formulation to perform autoregressive inference requires **constant memory** and is **many times faster**

# Summary

- **Kernel feature maps** and **matrix associativity** yield an attention with linear complexity.
- Computing the key value matrix as a **cumulative sum** extends our efficient attention computation to the autoregressive case
- Using the RNN formulation to perform autoregressive inference requires **constant memory** and is **many times faster**

# Thank you for your time!

Check out the code at https://linear-transformers.com/ .

```python
from fast_transformers.builders import TransformerEncoderBuilder
linear_bert = TransformerEncoderBuilder.from_kwargs(
    n_layers=12,
    n_heads=12,
    query_dimensions=64,
    value_dimensions=64,
    feed_forward_dimensions=3072,
    attention_type="linear",
).get()
# dummy 4000 long sequence
y = linear_bert(torch.rand(10, 4000, 768))
```

# References I

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

# References II


Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stker, and Alex Waibel. Self-attentional acoustic models. In *19th Annual Conference of the International Speech Communication Association (InterSpeech 2018)*, Hyderabad, India, September 2018.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

# References III


Niki Parmar, Prajit Ramachandran, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 68–80. Curran Associates, Inc., 2019.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*, 2020.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.