Efficient Transformers: Kernels and more

Angelos Katharopoulos

https://angeloskath.github.io/data/ml_collective_slides.pdf

ML Collective, October 30 2020

Funded by FNSNF

Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, François Fleuret

ICML 2020

A brief history of transformers

- Attention Is All You Need (NeurIPS 2017)

A brief history of transformers

- Attention Is All You Need (NeurIPS 2017)
- GPT (2018), XLNet (NeurIPS 2019) and BERT (NAACL 2019)

A brief history of transformers

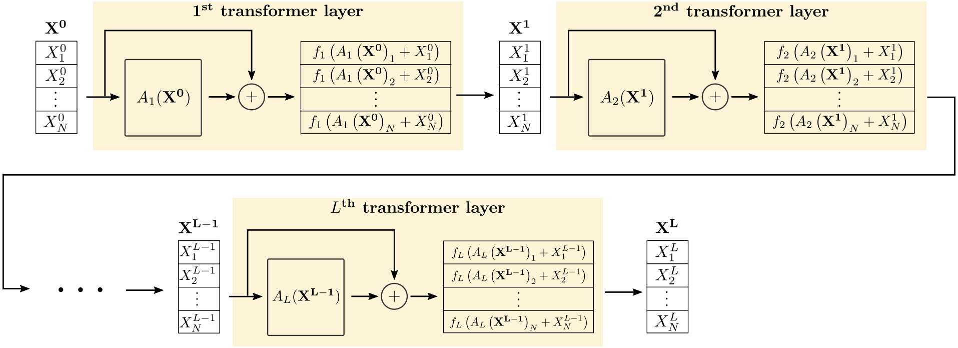
- Attention Is All You Need (NeurIPS 2017)
- GPT (2018), XLNet (NeurIPS 2019) and BERT (NAACL 2019)
- Image-GPT (ICML 2020), DETR (ECCV 2020) and Vision Transformer (ICLR 2021)
- Polygen (ICML 2020)
- Wav2Vec (NeurIPS 2020)

A brief history of transformers

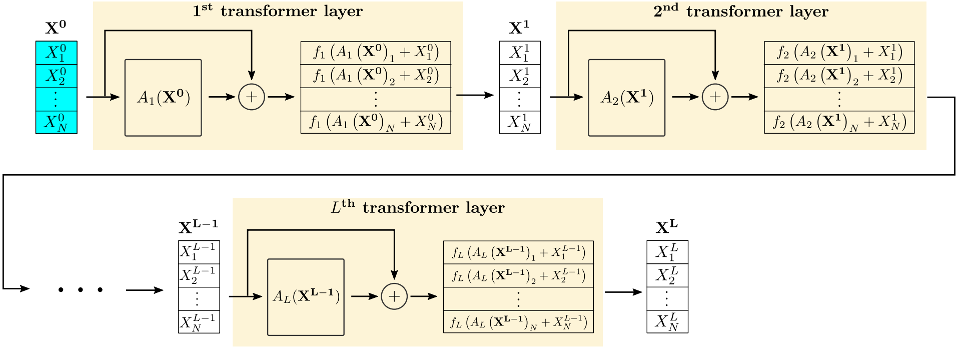
- Attention Is All You Need (NeurIPS 2017)
- GPT (2018), XLNet (NeurIPS 2019) and BERT (NAACL 2019)
- Image-GPT (ICML 2020), DETR (ECCV 2020) and Vision Transformer (ICLR 2021)
- Polygen (ICML 2020)
- Wav2Vec (NeurIPS 2020)

Transformers are related to Convolutional (Cordonnier et al., 2020), Recurrent (Katharopoulos et al., 2020) and Graph neural networks.

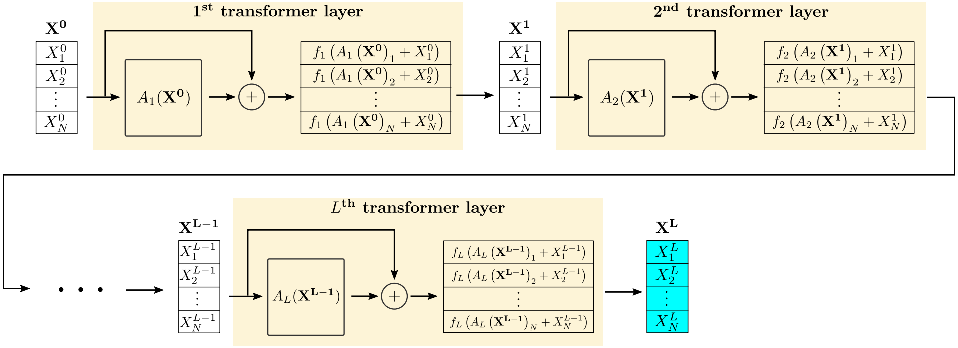
Definition of a transformer



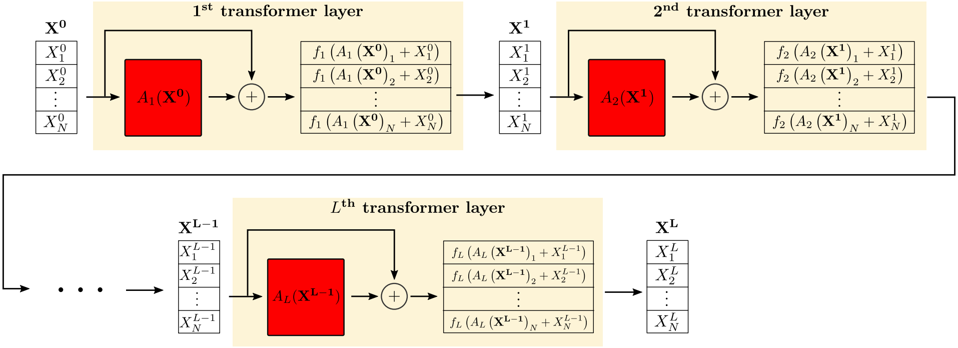
Definition of a transformer



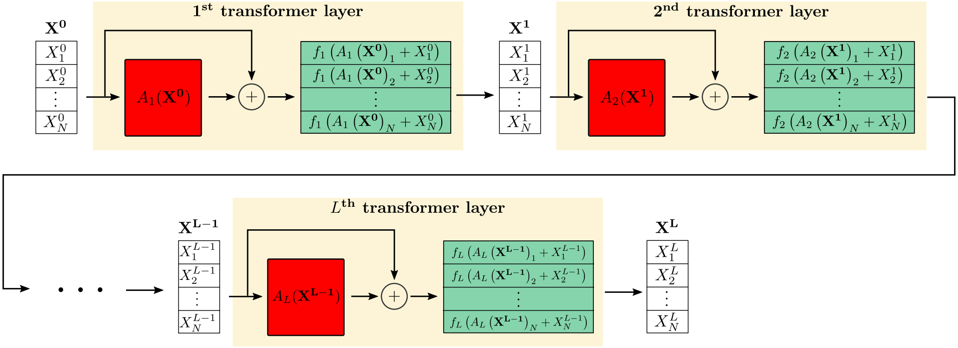
Definition of a transformer



Definition of a transformer

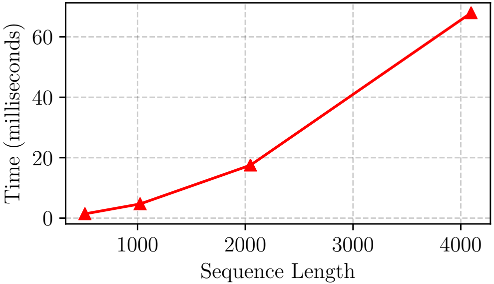


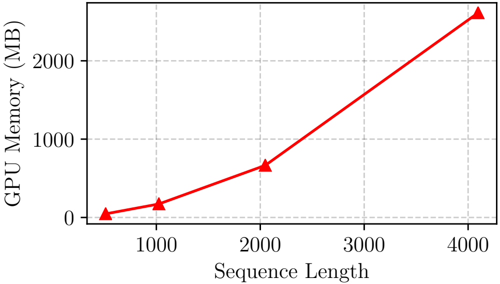
Definition of a transformer



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

Self-Attention

The commonly used attention mechanism is the scaled dot product attention

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

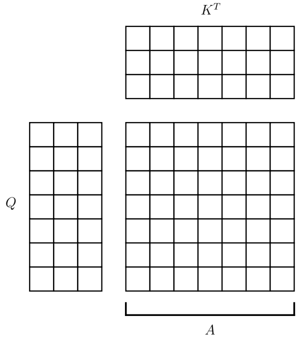
Self-Attention

The commonly used attention mechanism is the scaled dot product attention

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

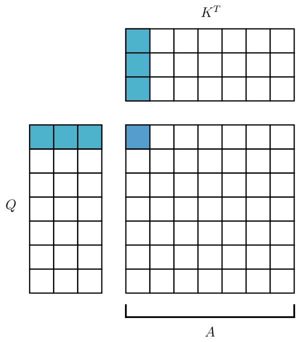
Quadratic complexity

Self-Attention



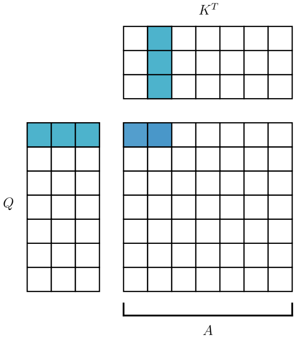
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



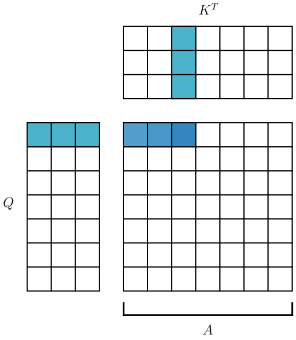
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



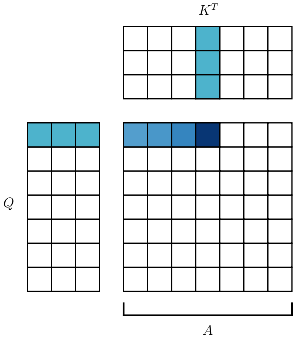
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



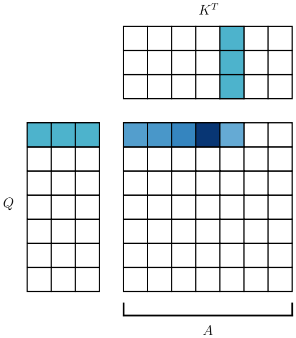
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



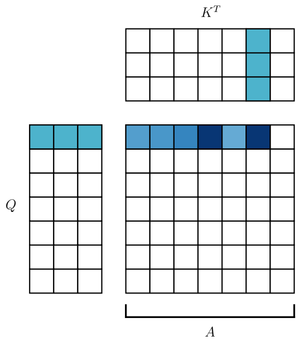
QK^T requires $\mathcal{O}\left(N^2D\right)$ multiplications and additions

Self-Attention



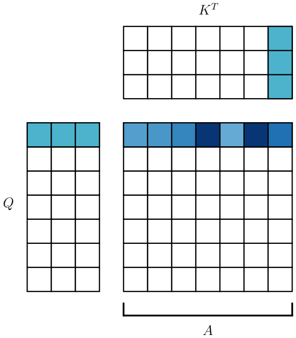
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



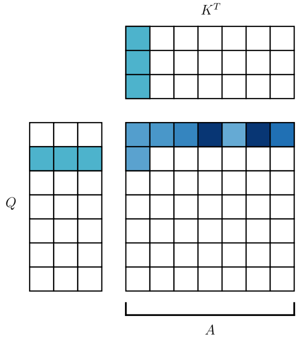
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



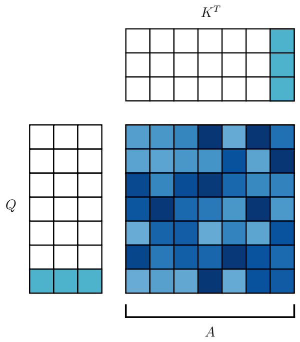
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



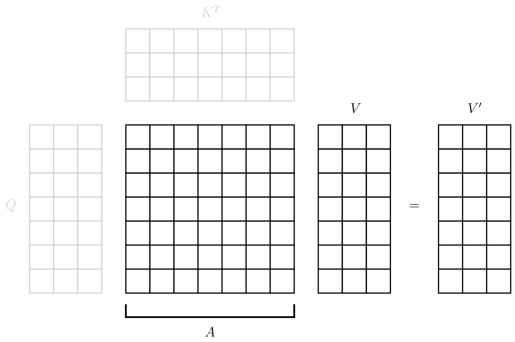
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



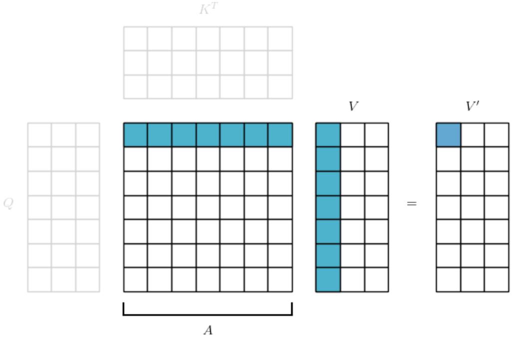
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



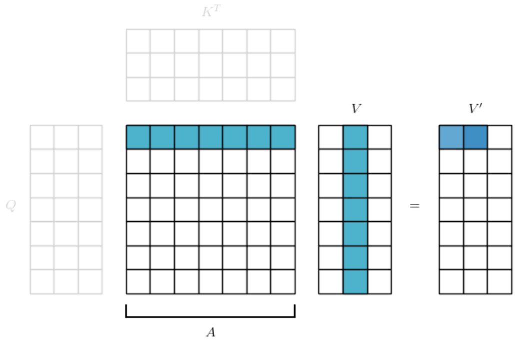
AV also requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



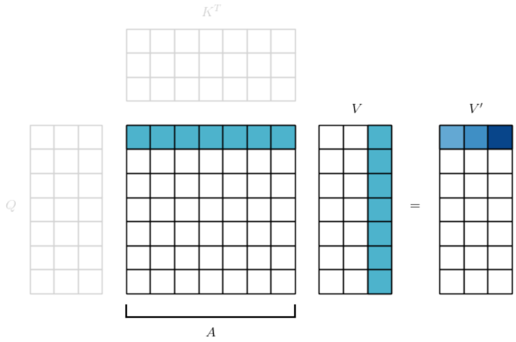
AV also requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



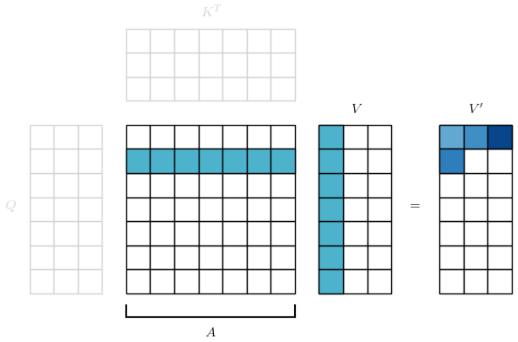
AV also requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



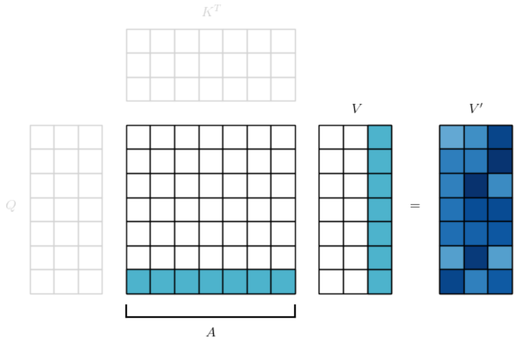
AV also requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



AV also requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



AV also requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Can we get rid of the $\mathcal{O}\left(N^2\right)$?

Can we get rid of the $\mathcal{O}\left(N^2\right)$?

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

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

Can we get rid of the $\mathcal{O}(N^2)$?

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

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

Kernelization

$$= \frac{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j) V_j}{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)}$$

Can we get rid of the $\mathcal{O}\left(N^2\right)$?

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

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

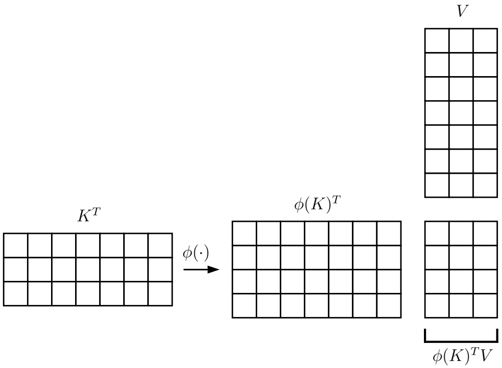
Kernelization

$$= \frac{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right) V_j}{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right)}$$

Associativity property

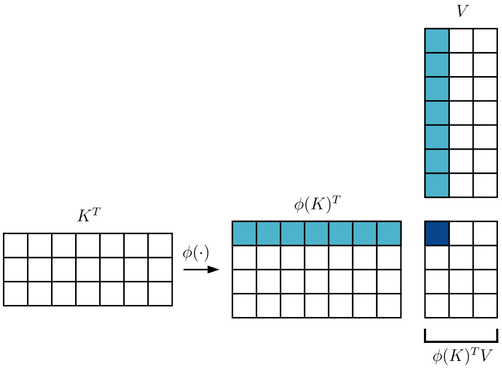
$$= \frac{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right) V_j^T}{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right)}$$

No explicit attention matrix



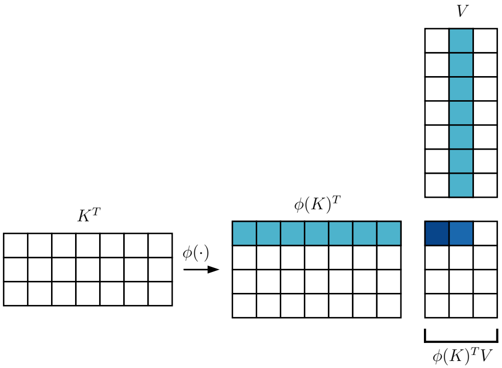
$\phi(K)^T V$ requires $\mathcal{O}(ND^2)$ multiplications and additions

No explicit attention matrix



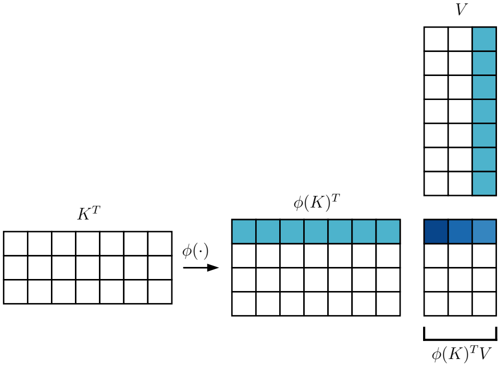
$\phi\left(K\right)^T V$ requires $\mathcal{O}\left(ND^2\right)$ multiplications and additions

No explicit attention matrix



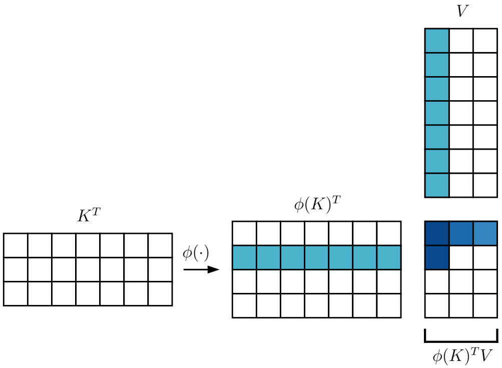
$\phi(K)^T V$ requires $\mathcal{O}(ND^2)$ multiplications and additions

No explicit attention matrix



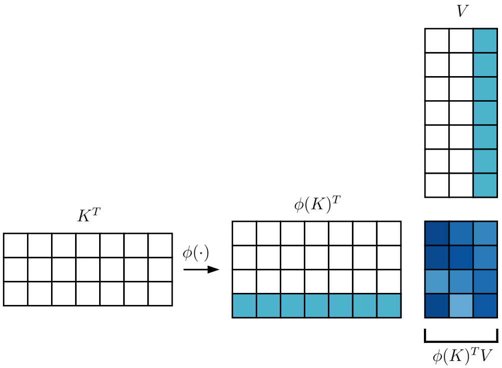
$\phi\,(K)^T\,V$ requires $\mathcal{O}\left(ND^2\right)$ multiplications and additions

No explicit attention matrix



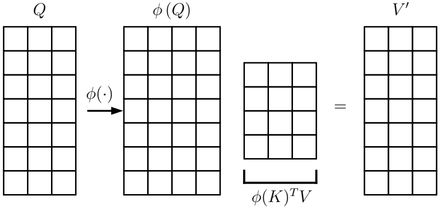
$\phi(K)^T V$ requires $\mathcal{O}(ND^2)$ multiplications and additions

No explicit attention matrix



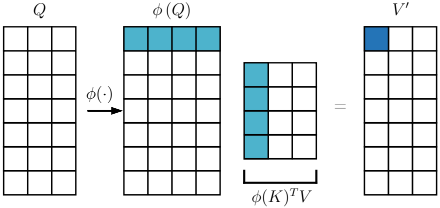
$\phi(K)^T V$ requires $\mathcal{O}(ND^2)$ multiplications and additions

No explicit attention matrix



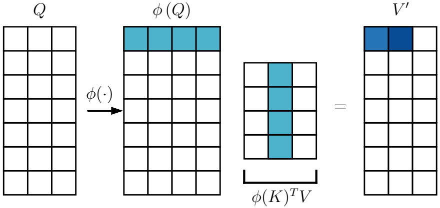
$V' = \phi(Q)\left(\phi(K)^T V\right)$ also requires $\mathcal{O}\left(ND^2\right)$ multiplications and additions

No explicit attention matrix



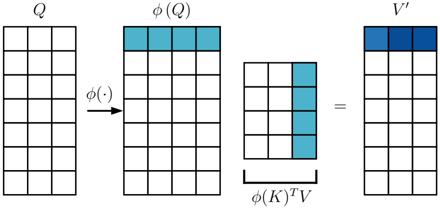
$V' = \phi(Q)\left(\phi(K)^T V\right)$ also requires $\mathcal{O}\left(ND^2\right)$ multiplications and additions

No explicit attention matrix



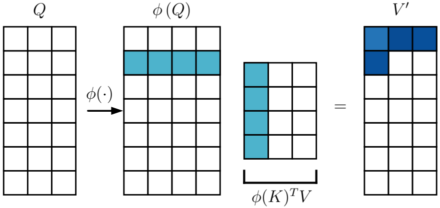
$V' = \phi(Q)\left(\phi(K)^T V\right)$ also requires $\mathcal{O}\left(ND^2\right)$ multiplications and additions

No explicit attention matrix



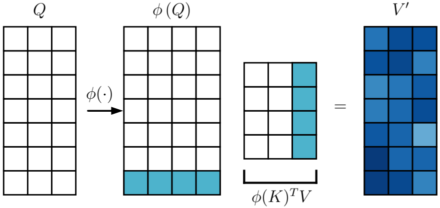
$V' = \phi(Q)\left(\phi(K)^T V\right)$ also requires $\mathcal{O}\left(ND^2\right)$ multiplications and additions

No explicit attention matrix



$V' = \phi(Q)\left(\phi(K)^T V\right)$ also requires $\mathcal{O}\left(ND^2\right)$ multiplications and additions

No explicit attention matrix



$V' = \phi(Q) \left(\phi(K)^T V \right)$ also requires $\mathcal{O}\left(ND^2\right)$ multiplications and additions

Causal Masking

Causal masking is used to efficiently train autoregressive transformers.

But we never compute the attention matrix! So what do we mask?

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

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

Autoregressive

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

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

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

Autoregressive

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

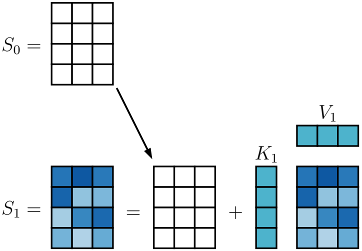
Naive computation of S_i and Z_i results in quadratic complexity.

Causal Masking

$$S_0 = \begin{array}{|c|c|c|}\hline & & \\ \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

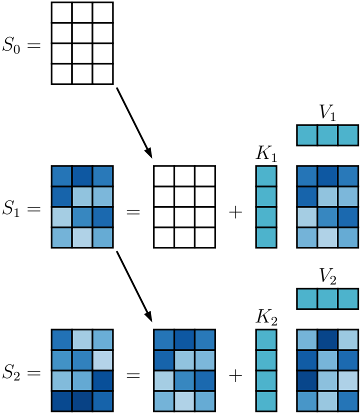
S_i and Z_i is an intermediate state that can be computed in $\mathcal{O}(1)$ from S_{i-1} and Z_{i-1}.

Causal Masking



S_i and Z_i is an intermediate state that can be computed in $\mathcal{O}(1)$ from S_{i-1} and Z_{i-1}.

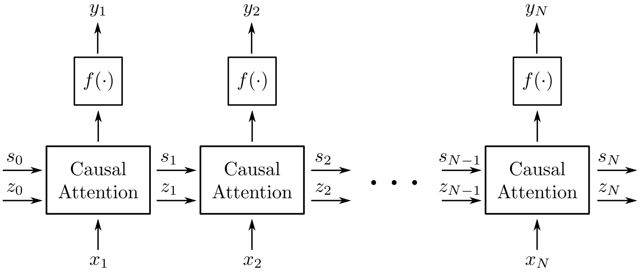
Causal Masking



S_i and Z_i is an intermediate state that can be computed in $\mathcal{O}(1)$ from S_{i-1} and Z_{i-1}.

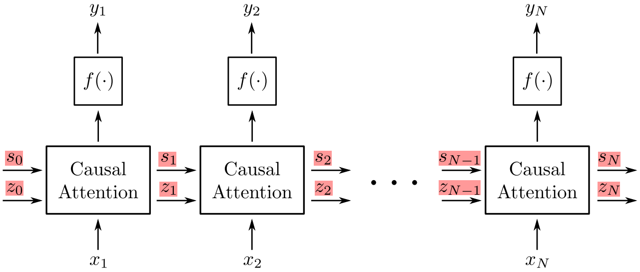
Transformers are RNNs

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.



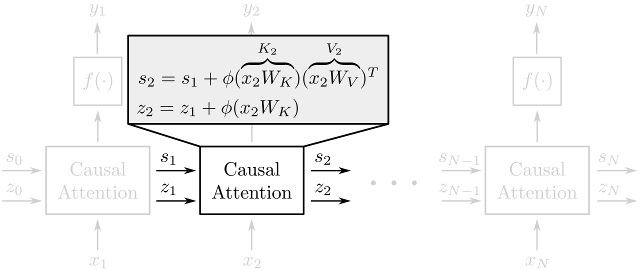
Transformers are RNNs

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.



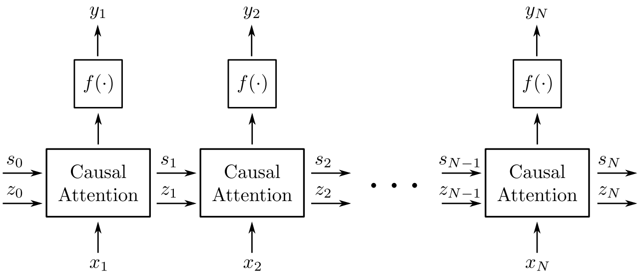
Transformers are RNNs

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 (1)

Our theoretical analysis holds for all transformers that use a similarity score that can be written as a kernel.

- Performers (Choromanski et al., 2020) recently introduced random Fourier features specifically tailored for this application.
- Simpler feature maps that do not correspond to any obvious kernel are good enough most times.
- There is a direct tradeoff between expressivity and computation time by increasing the dimensionality of the features.

Practical implications (2)

The gradients of causally masked transformers can be formulated in $\mathcal{O}(ND)$ space and $\mathcal{O}\left(ND^2\right)$ time.

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

Autograd needs to keep S_i in memory $\forall\, i$.

Code availability

PyTorch code available at https://github.com/idiap/fast-transformers.

```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))
```

Experimental setup

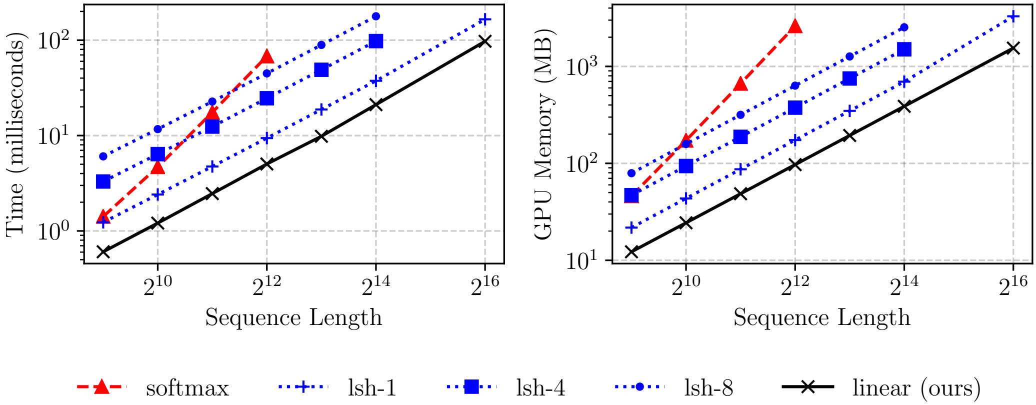
Baselines

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

Autoregressive image generation

- Generative modeling of images byte by byte
- We use discretized mixture of logistics to model the pixel
- MNIST and CIFAR have sequence lengths 784 and 3,072 respectively

Autoregressive image generation

Unconditional samples after 250 epochs on MNIST

Ours (0.644 bpd)

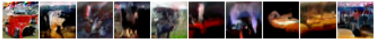
Softmax (0.621 bpd)

LSH-1 (0.745 bpd)

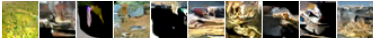
LSH-4 (0.676 bpd)

Unconditional samples after 1 GPU week on CIFAR-10

Ours (3.40 bpd)

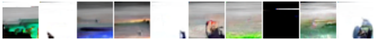


Softmax (3.47 bpd)

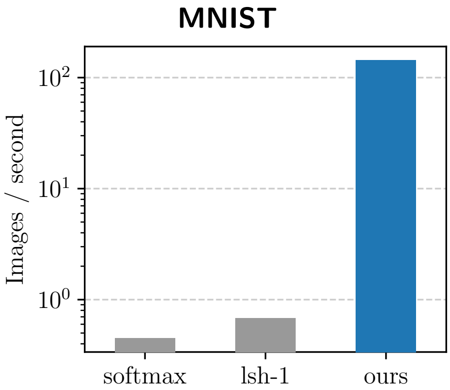


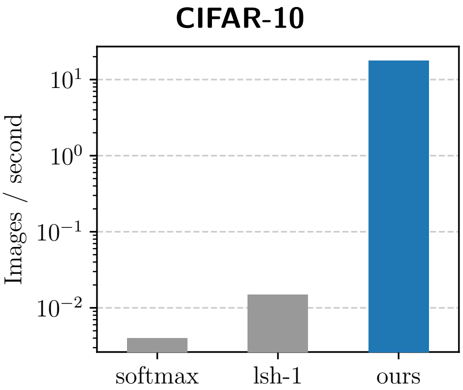
LSH-1 (3.39 bpd)

LSH-4 (3.51 bpd)

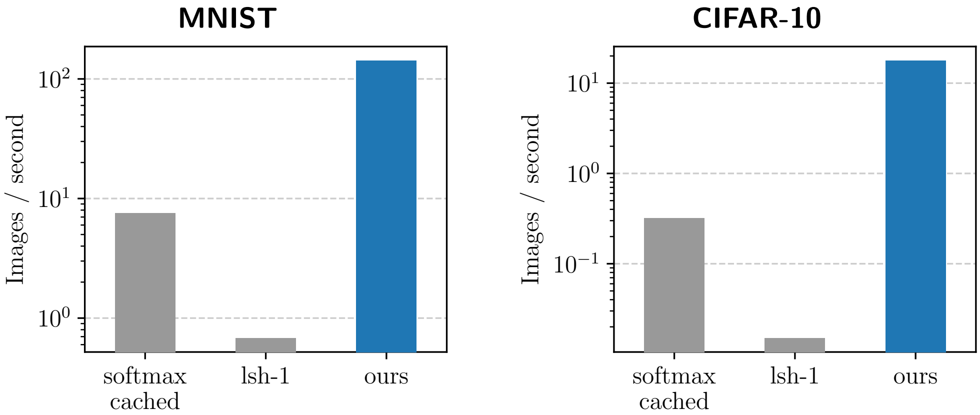


Autoregressive image generation throughput

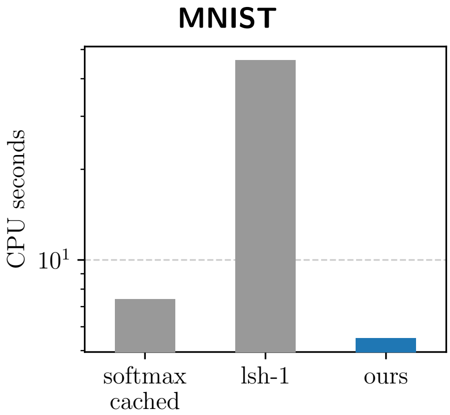


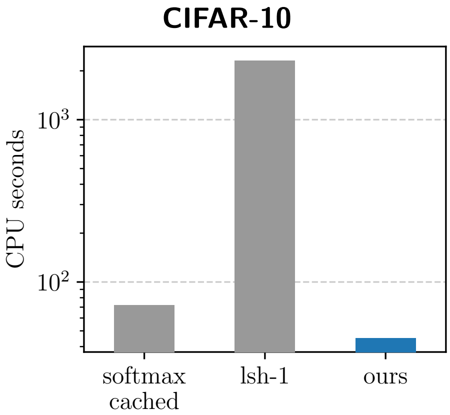


Autoregressive image generation throughput



Autoregressive image generation latency

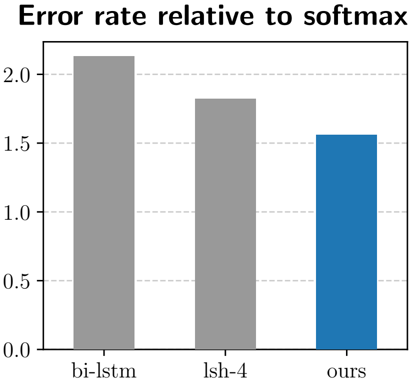




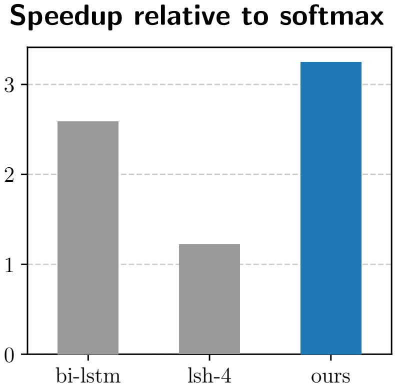
Automatic speech recognition

- Classification of a sequence of features to phonemes
- Variable length sequences with an average length of 800 and a maximum of 2,400
- We also compare with a commonly used bidirectional LSTM baseline

Automatic speech recognition



Lower is better

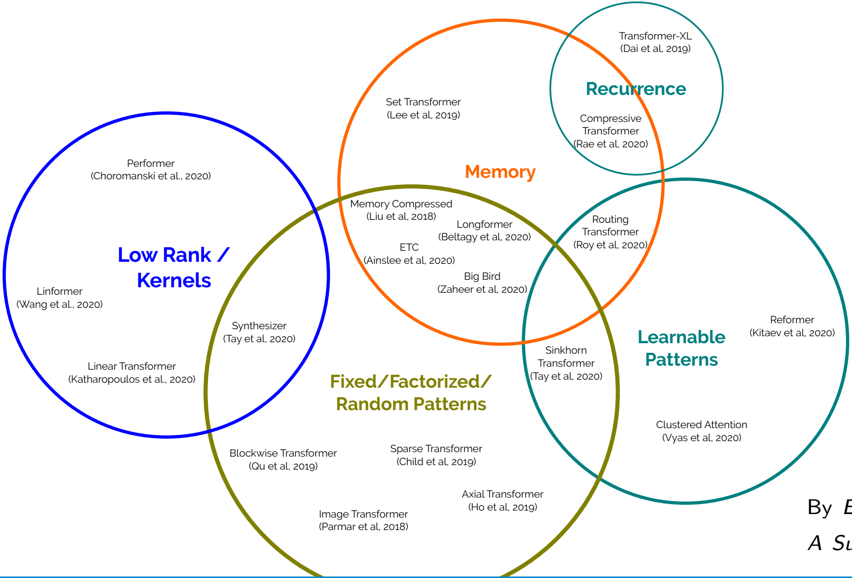


Higher is better

Summary

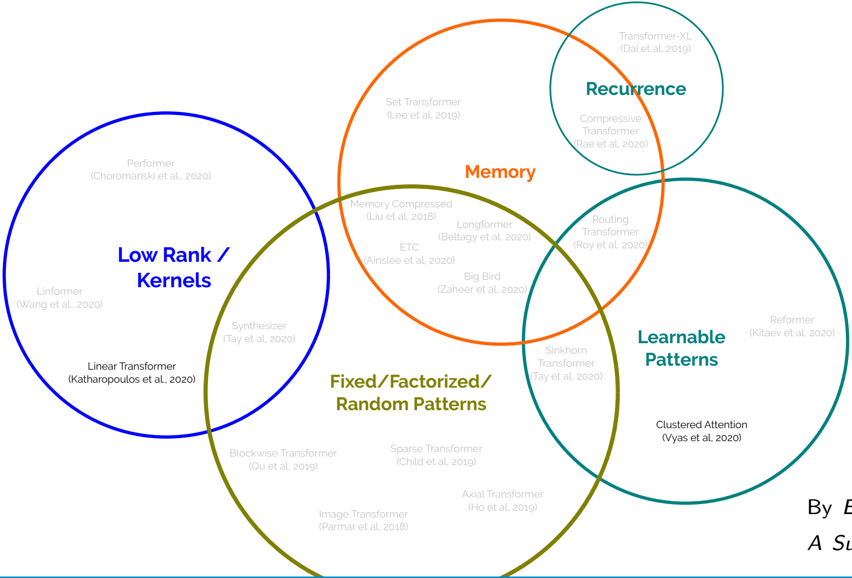
- **Kernel feature maps** and **matrix associativity** yield an attention with linear complexity.
- Computing the key value matrix as a **cumulative sum** extends our efficient attention computation to the autoregressive case
- Using the RNN formulation to perform autoregressive inference requires **constant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

Where does this work fit in?



By *Efficient Transformers: A Survey* (Tay et al., 2020)

Where does this work fit in?



By *Efficient Transformers: A Survey* (Tay et al., 2020)

Fast Transformers with Clustered Attention

Apoorv Vyas, Angelos Katharopoulos, François Fleuret

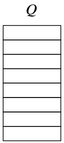
To appear in NeurIPS 2020

Softmax approximation

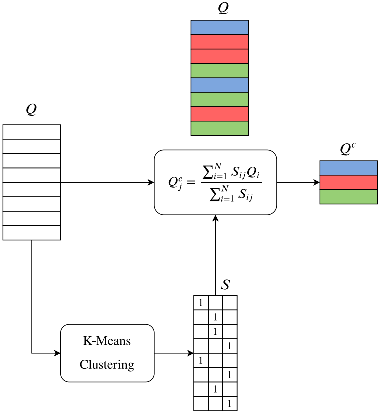
Given Q_i and Q_j such that $\|Q_i - Q_j\|_2 \leq \epsilon$ then

$$\|\,\mathrm{softmax}\left(Q_i K^T\right) - \mathrm{softmax}\left(Q_j K^T\right)\|_2 \leq \epsilon \|K\|_2$$

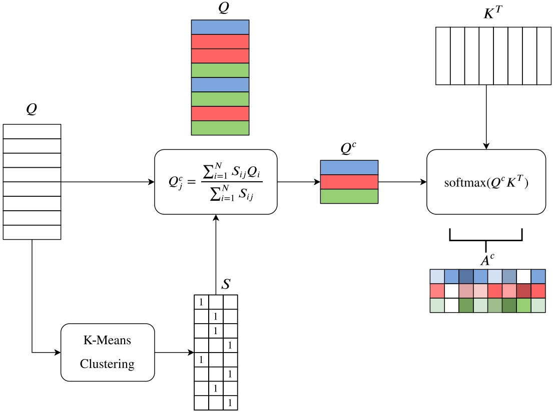
Clustered attention



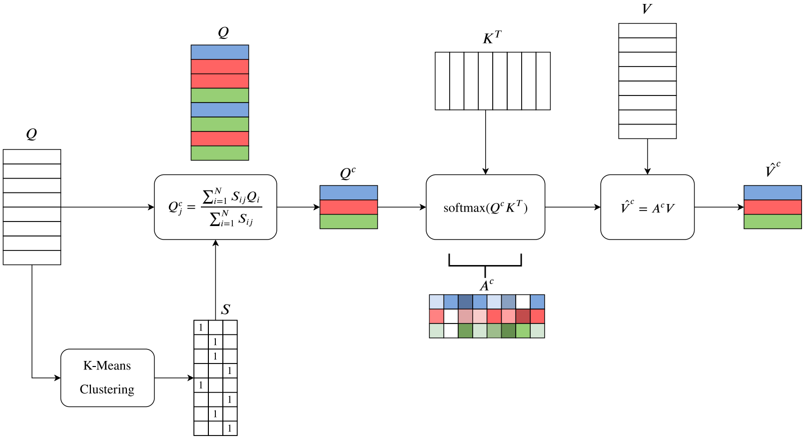
Clustered attention



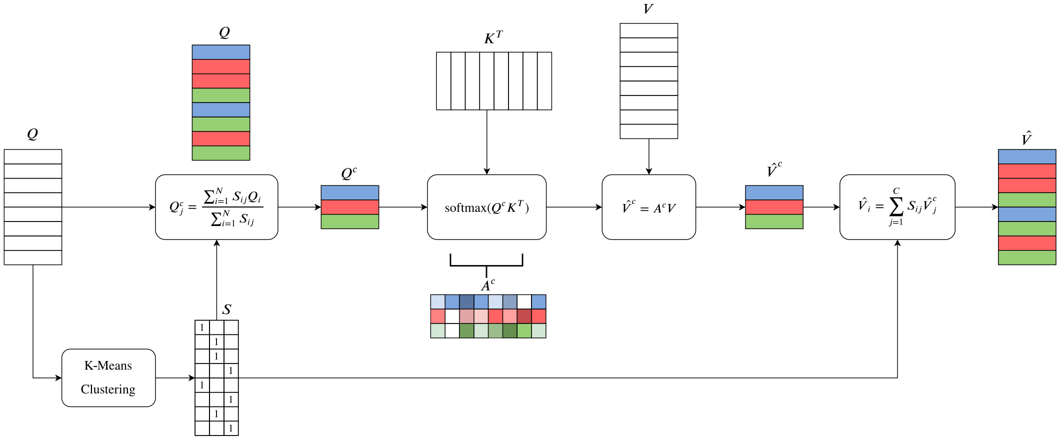
Clustered attention



Clustered attention



Clustered attention

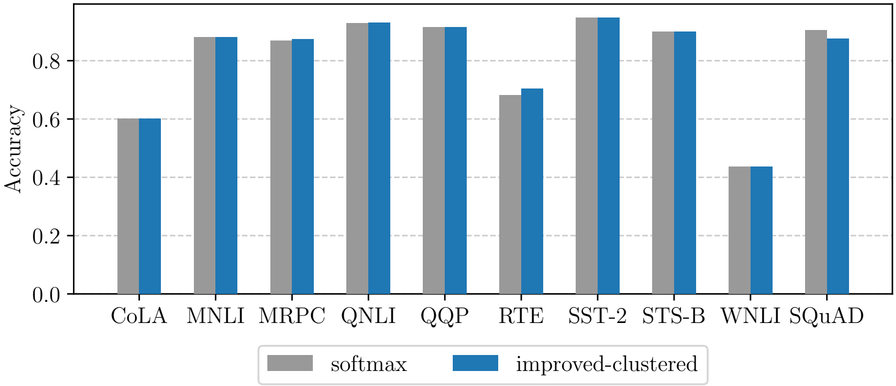


Improved clustered attention

- The approximation can be improved by computing any dot products exactly
- We select the top-k dot products per query cluster
- Selecting groups of keys results in efficient GPU implementations

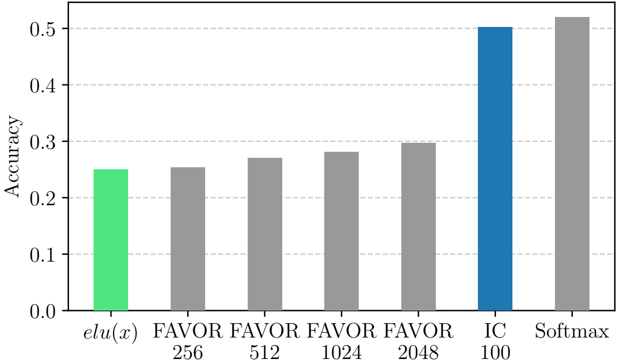
RoBERTa approximation

RoBERTa approximation on GLUE and SQUAD benchmarks with **25 clusters**.



Wav2Vec approximation

Wav2Vec approximation on LibriSpeech.



Thank you for your time!

https://github.com/idiap/fast-transformers

References I

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

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020. URL https://arxiv.org/pdf/2006.16236.pdf.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

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.

References II

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

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020.