Efficient Transformers: Kernels and Clustering

Angelos Katharopoulos

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

AVG-RG, January 29th 2021

Funded by FNSNF

A brief history of transformers

- Attention Is All You Need (NeurIPS 2017)

A brief history of transformers

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

A brief history of transformers

- Attention Is All You Need (NeurIPS 2017)
- **Language modeling**: GPT (2018), XLNet (NeurIPS 2019) and BERT (NAACL 2019)
- **Image processing**: Image-GPT (ICML 2020), DETR (ECCV 2020) and ViT (ICLR 2021)
- **3D Vision**: Polygen (ICML 2020), PCT (2021)
- **Audio**: Music Transformer (ICLR 2019), Wav2Vec (NeurIPS 2020)

A brief history of transformers

- Attention Is All You Need (NeurIPS 2017)
- **Language modeling**: GPT (2018), XLNet (NeurIPS 2019) and BERT (NAACL 2019)
- **Image processing**: Image-GPT (ICML 2020), DETR (ECCV 2020) and ViT (ICLR 2021)
- **3D Vision**: Polygen (ICML 2020), PCT (2021)
- **Audio**: Music Transformer (ICLR 2019), Wav2Vec (NeurIPS 2020)

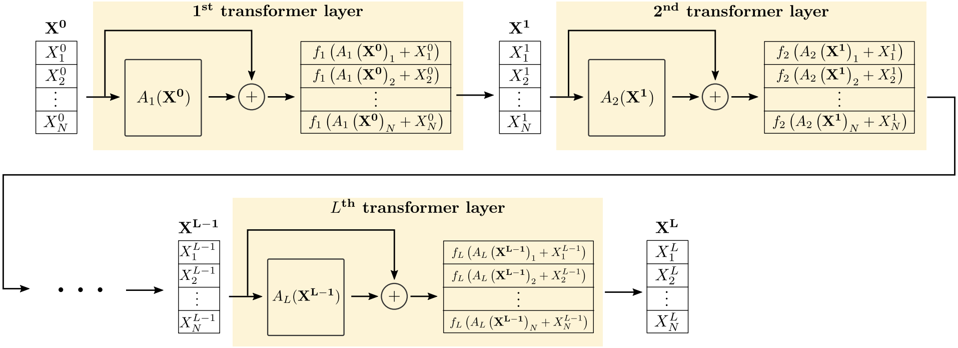
Transformers are related to Convolutional (Cordonnier et al., 2020), Recurrent (Katharopoulos et al., 2020) and Graph neural networks.

Motivation for transformers

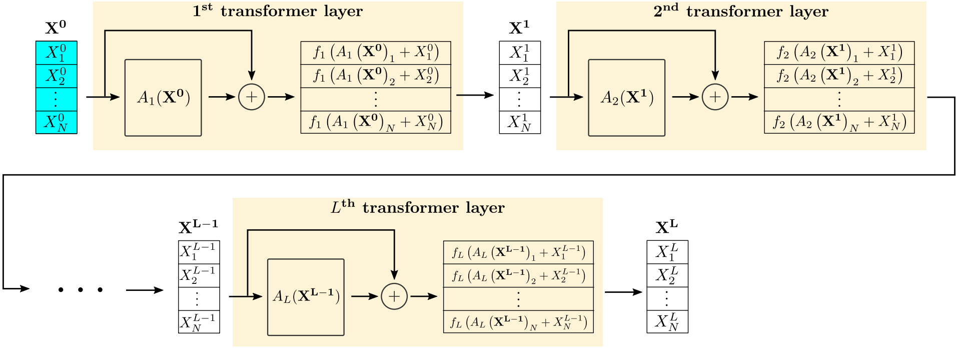
The transformer architecture tackles the two main problems of RNNs

- Forgetting information from previous inputs
- Parallelization of the computation

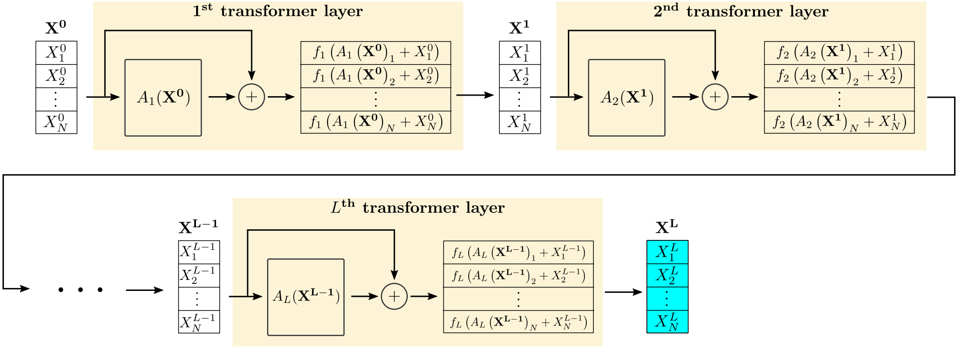
Definition of a transformer



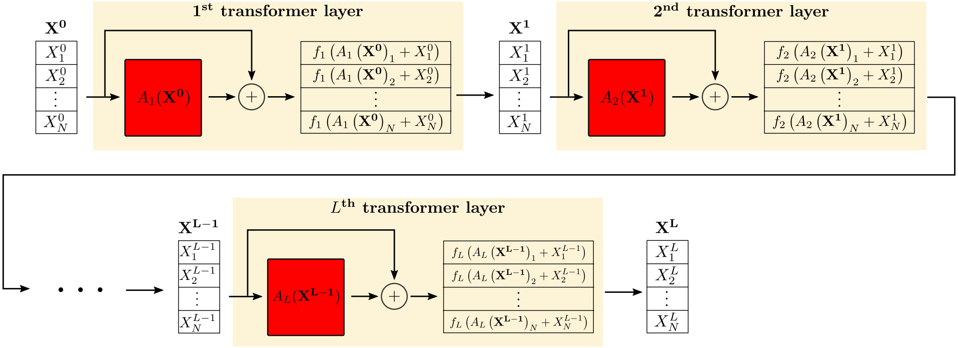
Definition of a transformer



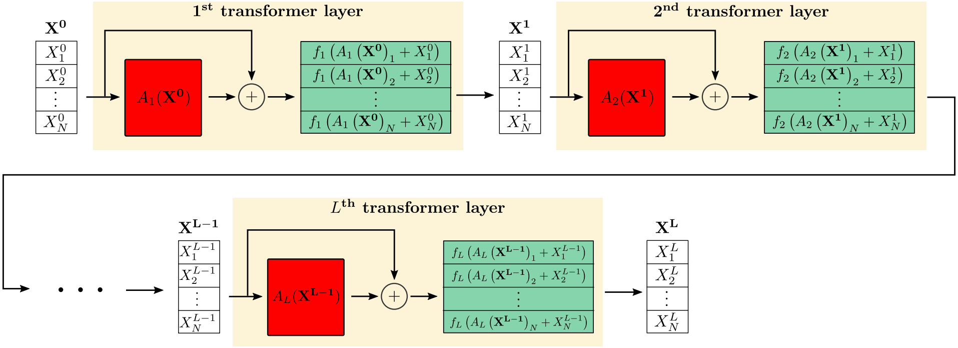
Definition of a transformer



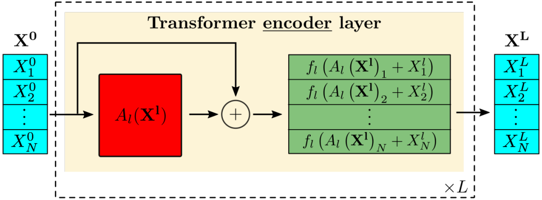
Definition of a transformer



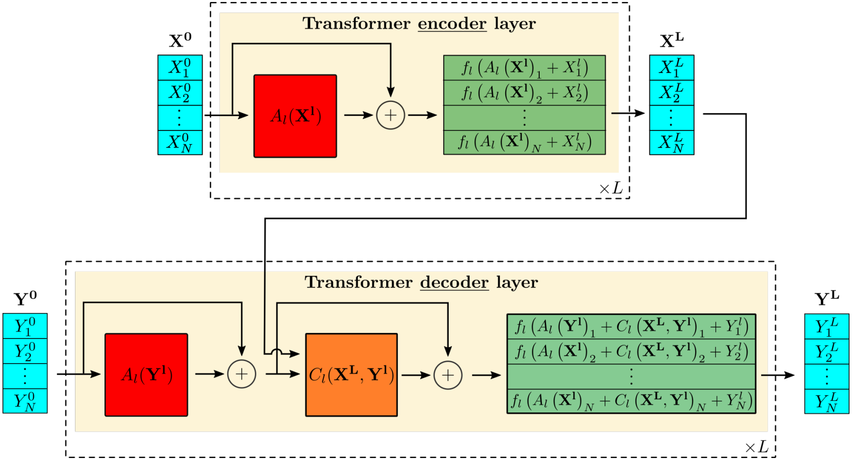
Definition of a transformer



Transformer Encoder/Decoder



Transformer Encoder/Decoder



Self-Attention

The commonly used attention mechanism is the scaled dot product attention

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

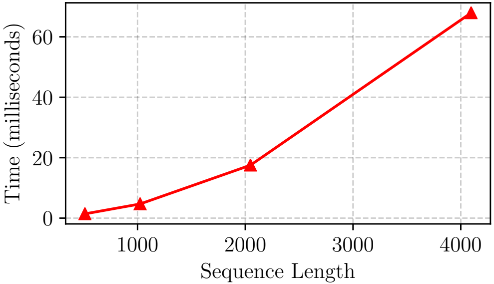
Cross-Attention

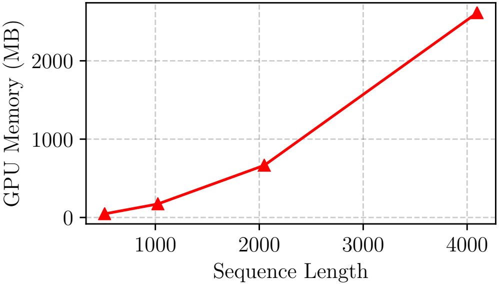
The only difference with cross-attention is the source of queries and keys.

$$
\begin{aligned}
Q &= Y W_Q \\
K &= X W_K \\
V &= X W_V \\
C_l(X, Y) = V' &= \operatorname{softmax}\left(\frac{QK^T}{\sqrt{D}}\right) V
\end{aligned}
$$

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

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

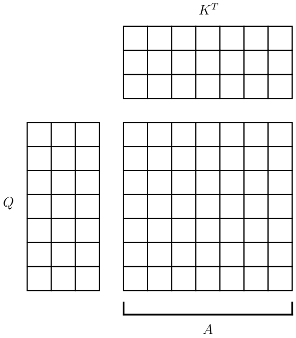
Self-Attention

The commonly used attention mechanism is the scaled dot product attention

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

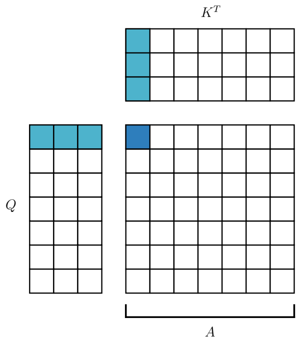
Quadratic complexity

Self-Attention



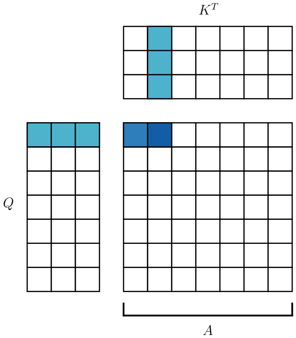
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



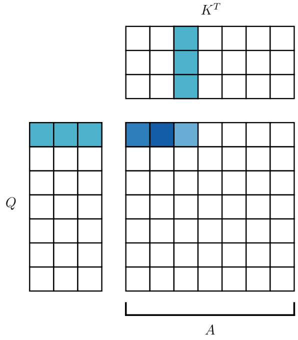
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



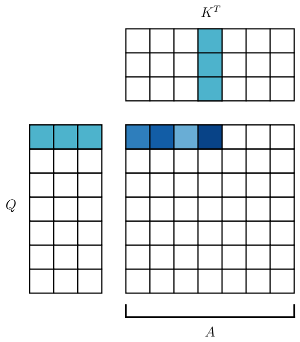
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



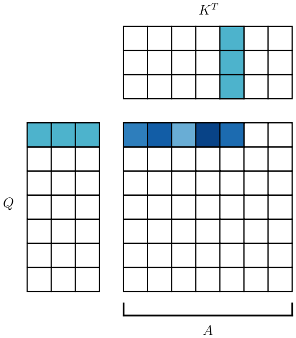
QK^T requires $\mathcal{O}\left(N^2D\right)$ multiplications and additions

Self-Attention



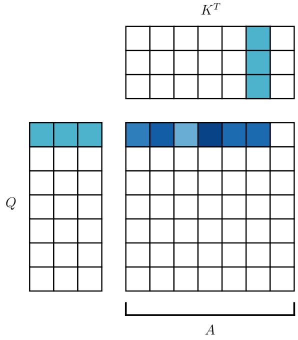
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



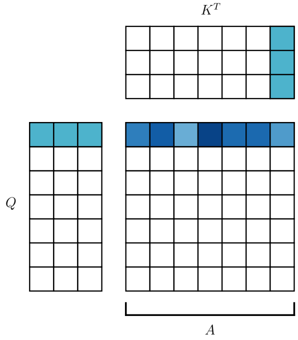
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



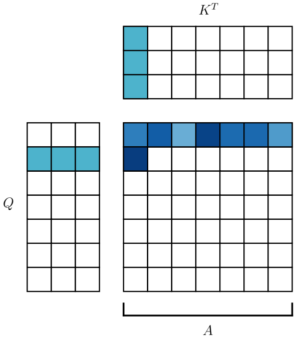
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



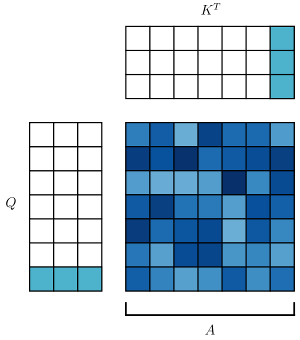
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



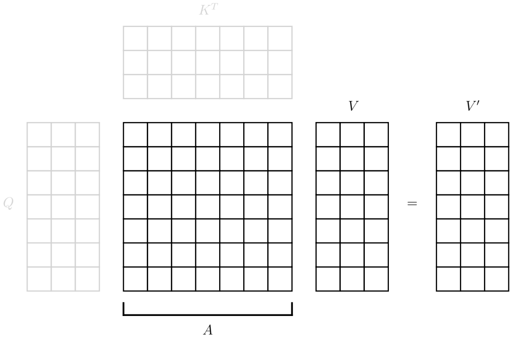
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



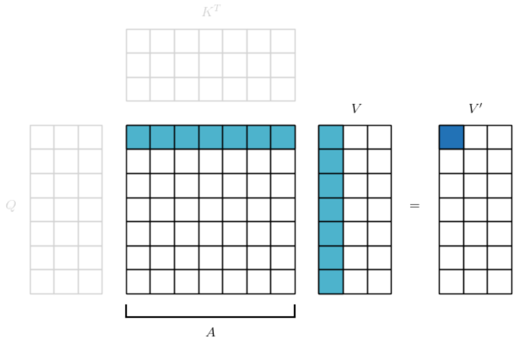
QK^T requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



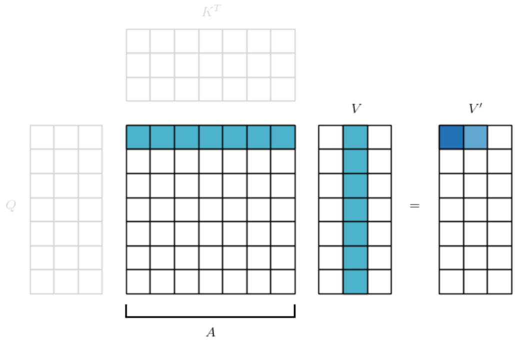
AV also requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



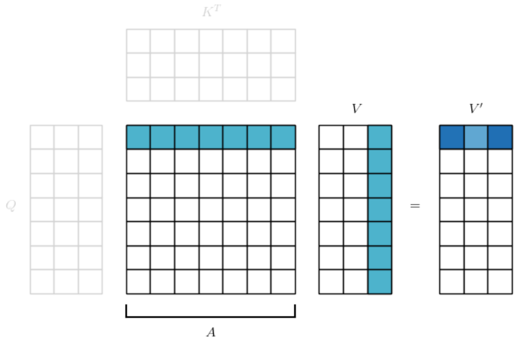
AV also requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



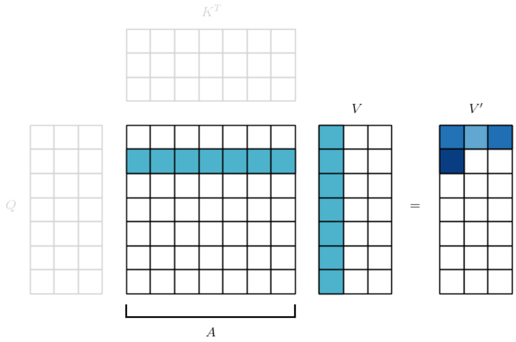
AV also requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

Self-Attention



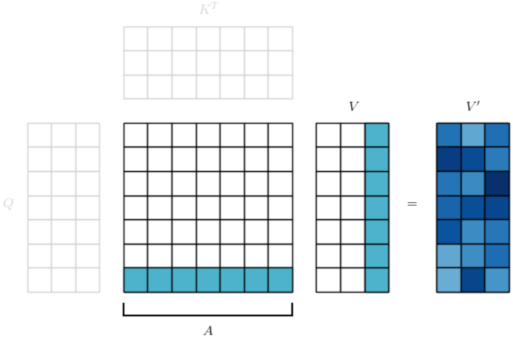
AV also requires $\mathcal{O}\left(N^2 D\right)$ multiplications and additions

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

Transformers are RNNs:
Fast Autoregressive Transformers with Linear Attention

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, François Fleuret

ICML 2020

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

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

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

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

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

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

Kernelization

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

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

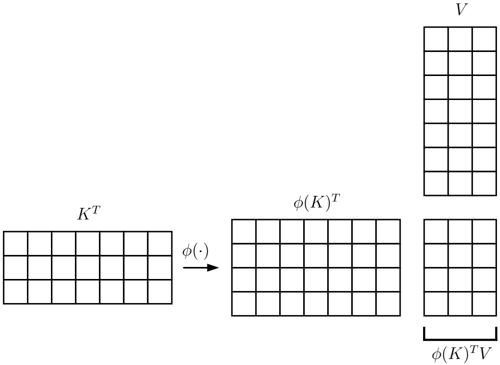
Matrix products are associative which makes the attention computation $\mathcal{O}\left(N\right)$ with respect to the sequence length.

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

Kernelization

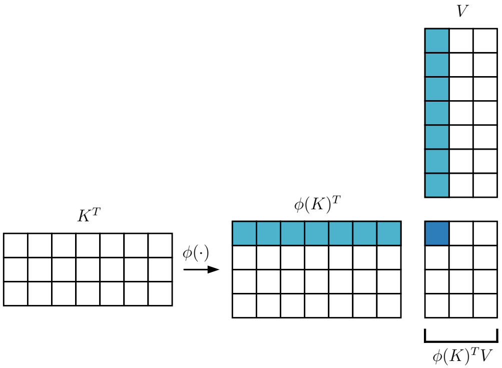
Associativity property

No explicit attention matrix



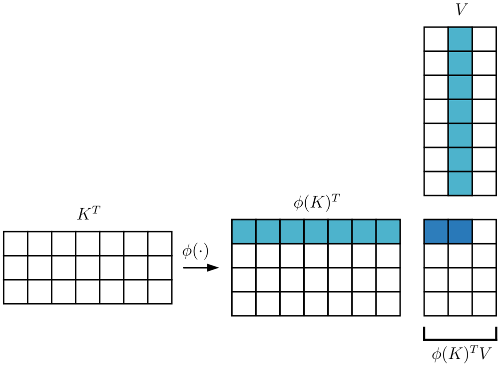
$\phi(K)^T V$ requires $\mathcal{O}(ND^2)$ multiplications and additions

No explicit attention matrix



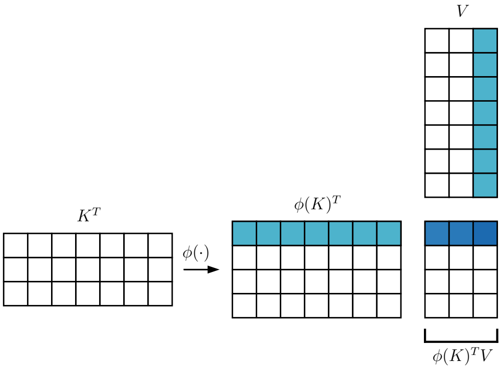
$\phi(K)^T V$ requires $\mathcal{O}(ND^2)$ multiplications and additions

No explicit attention matrix



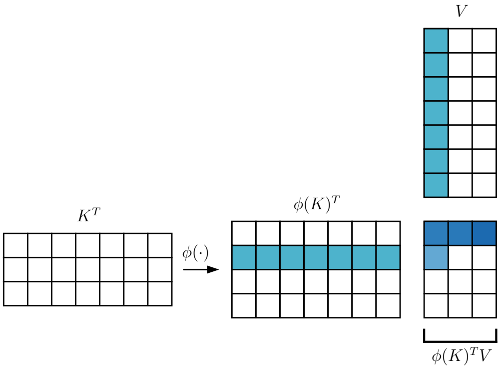
$\phi(K)^T V$ requires $\mathcal{O}(ND^2)$ multiplications and additions

No explicit attention matrix



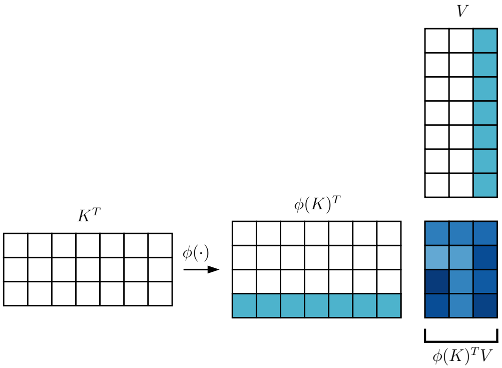
$\phi(K)^T V$ requires $\mathcal{O}(ND^2)$ multiplications and additions

No explicit attention matrix



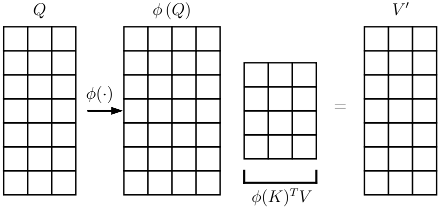
$\phi(K)^T V$ requires $\mathcal{O}(ND^2)$ multiplications and additions

No explicit attention matrix



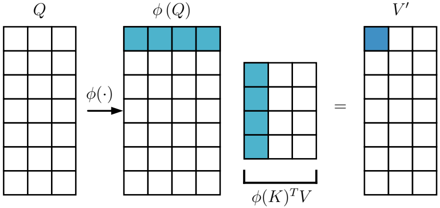
$\phi\left(K\right)^T V$ requires $\mathcal{O}\left(ND^2\right)$ multiplications and additions

No explicit attention matrix



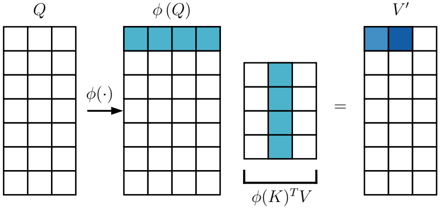
$V' = \phi(Q)\left(\phi(K)^T V\right)$ also requires $\mathcal{O}\left(ND^2\right)$ multiplications and additions

No explicit attention matrix



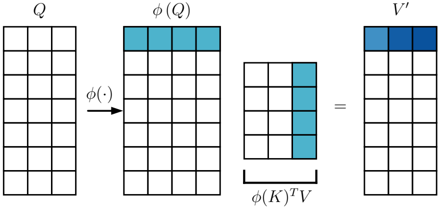
$V' = \phi(Q)\left(\phi(K)^T V\right)$ also requires $\mathcal{O}\left(ND^2\right)$ multiplications and additions

No explicit attention matrix



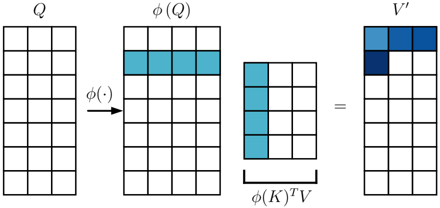
$V' = \phi(Q)\left(\phi(K)^T V\right)$ also requires $\mathcal{O}\left(ND^2\right)$ multiplications and additions

No explicit attention matrix



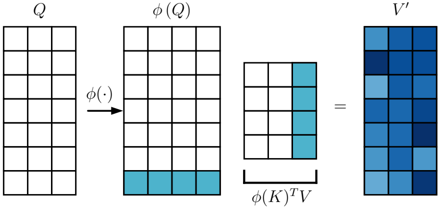
$V' = \phi(Q)\left(\phi(K)^T V\right)$ also requires $\mathcal{O}\left(ND^2\right)$ multiplications and additions

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

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

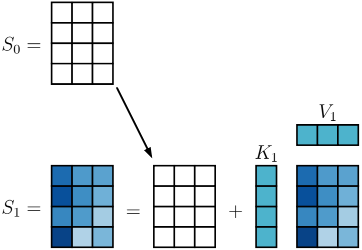
Naive computation of S_i and Z_i results in quadratic complexity.

Causal Masking

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

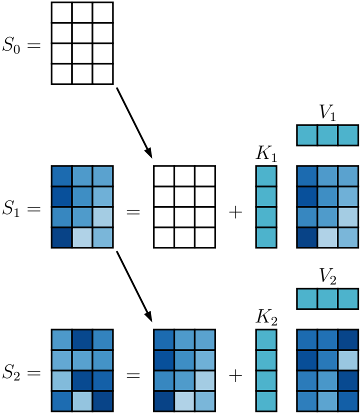
S_i and Z_i is an intermediate state that can be computed in $\mathcal{O}(1)$ from S_{i-1} and Z_{i-1}.

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}.

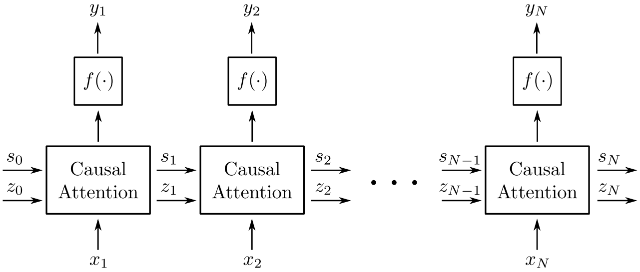
Causal Masking



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}.

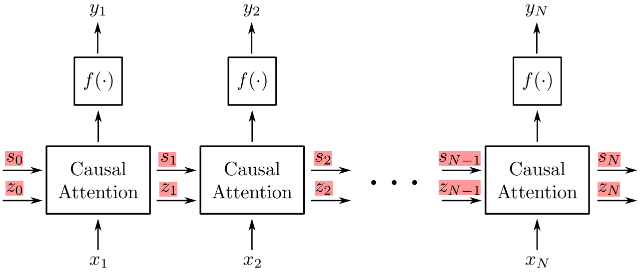
Transformers are RNNs

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.



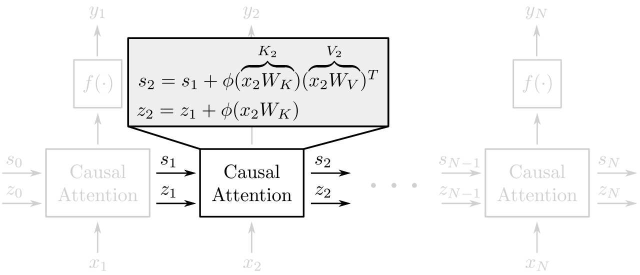
Transformers are RNNs

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.



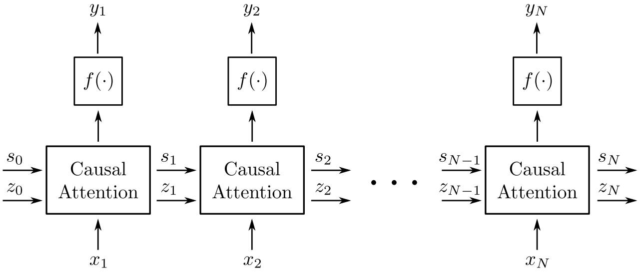
Transformers are RNNs

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 $\mathcal{O}(ND^2)$ time.

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

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

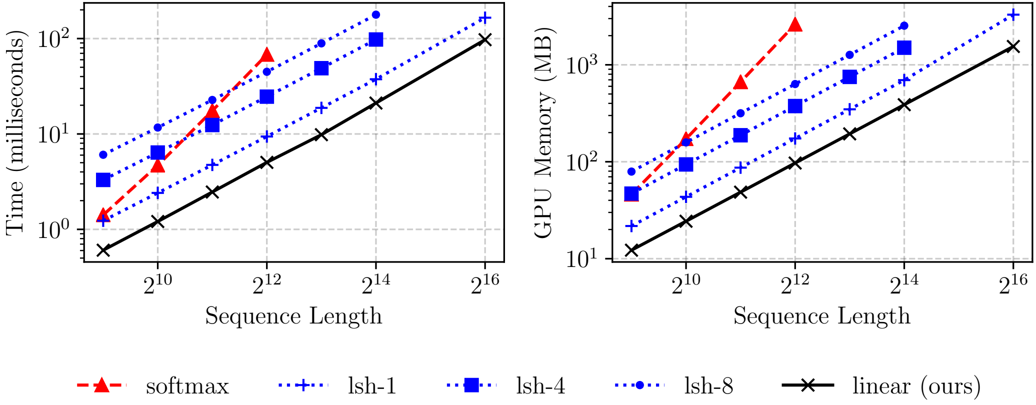
Baselines

- 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

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)

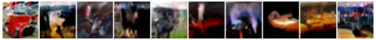
Softmax (0.621 bpd)

LSH-1 (0.745 bpd)

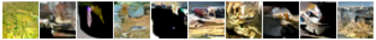
LSH-4 (0.676 bpd)

Unconditional samples after 1 GPU week on CIFAR-10

Ours (3.40 bpd)

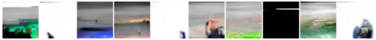


Softmax (3.47 bpd)

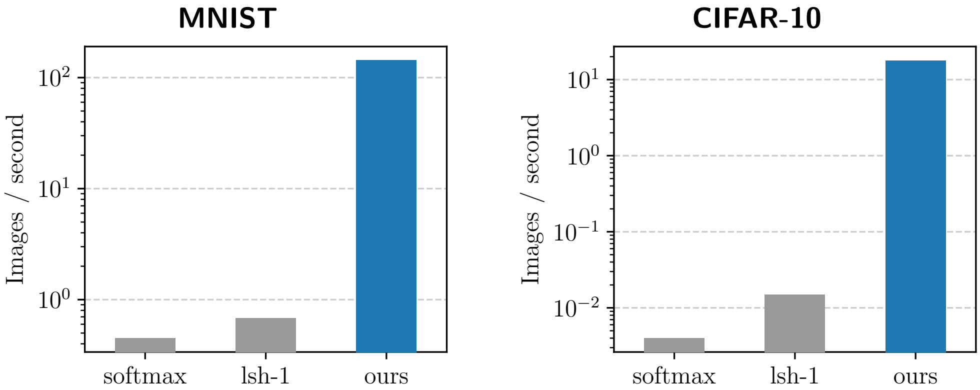


LSH-1 (3.39 bpd)

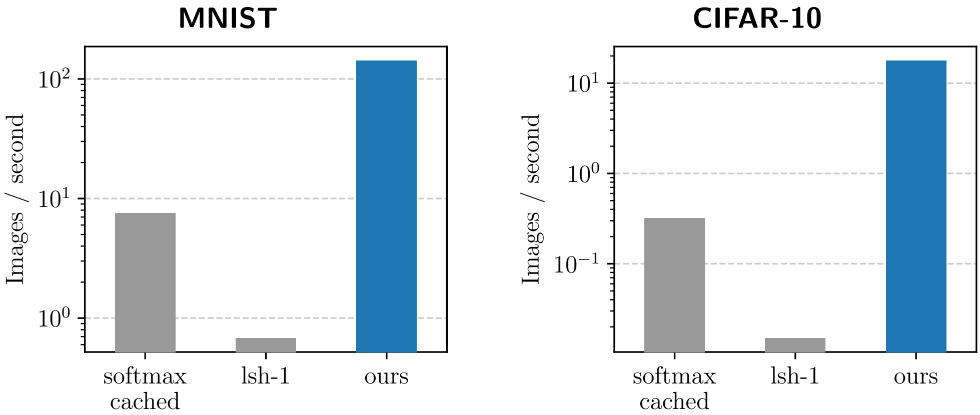
LSH-4 (3.51 bpd)



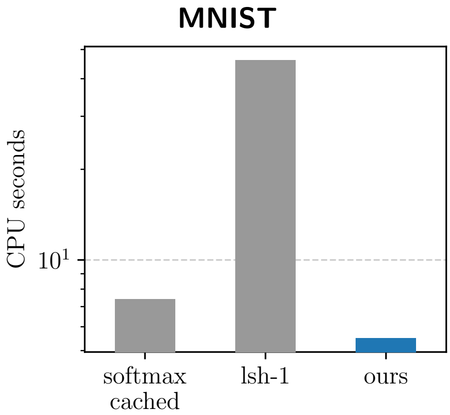
Autoregressive image generation throughput

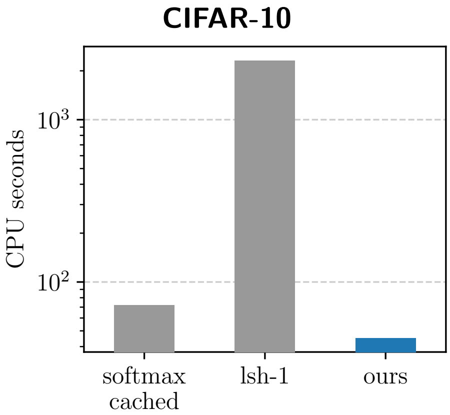


Autoregressive image generation throughput



Autoregressive image generation latency





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**

Caveats

- This is not a silver bullet! To get the speed we have to give up something...
  The attention matrix is no longer full rank!

Caveats

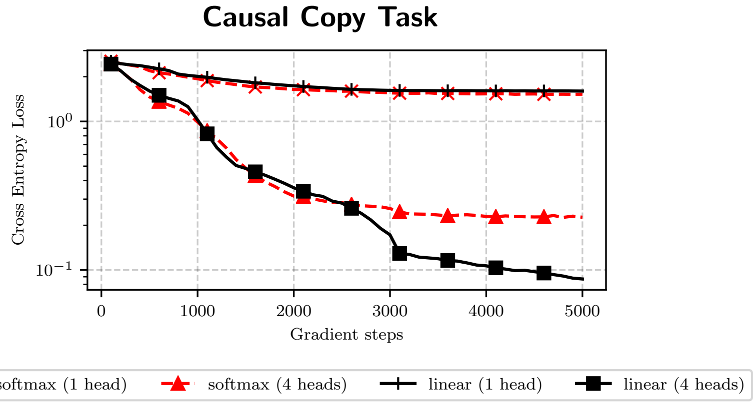
- This is not a silver bullet! To get the speed we have to give up something... **The attention matrix is no longer full rank!**
- The training dynamics can be different. Do we need different optimizers?

Do we need full rank?

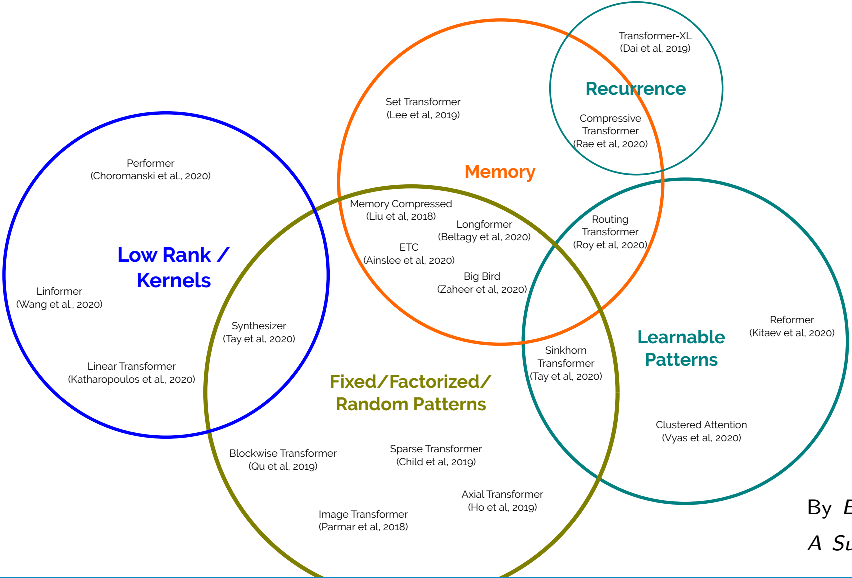
Can we learn to copy a sequence of length 32 with

- a 16 dimensional feature map
- a single layer single head transformer

Do we need full rank?

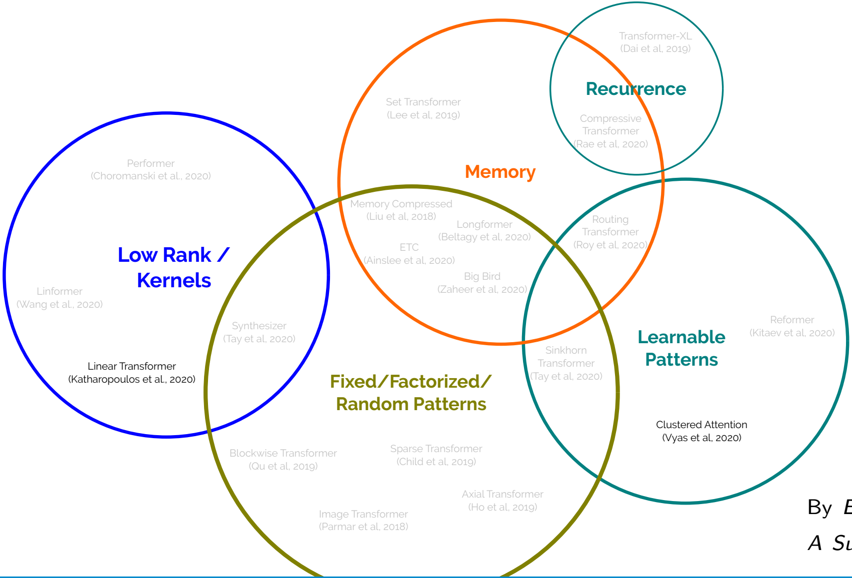


Where does this work fit in?



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

NeurIPS 2020

Fast Transformers with Clustered Attention

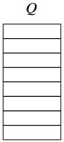
- A fast **approximation of self-attention** by clustering the queries
- Linear computational and memory complexity for a fixed number of clusters
- Approximation of pretrained transformers **without finetuning and without loss in performance**

Softmax approximation

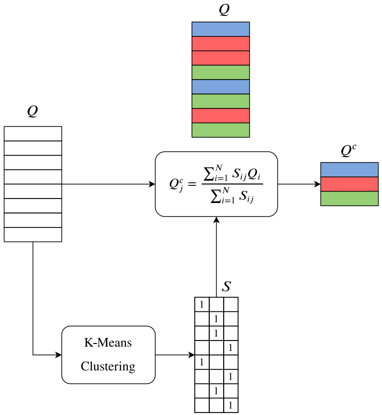
Given Q_i and Q_j such that $\|Q_i - Q_j\|_2 \leq \epsilon$ then

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

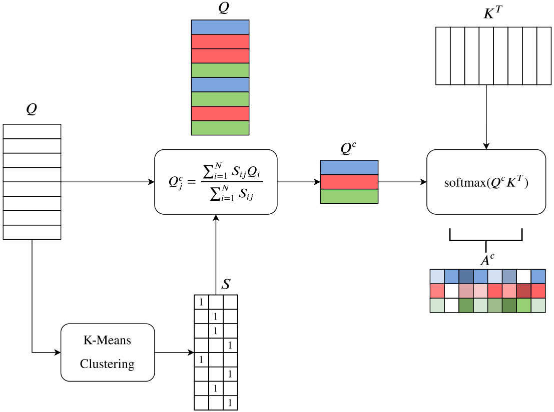
Clustered attention



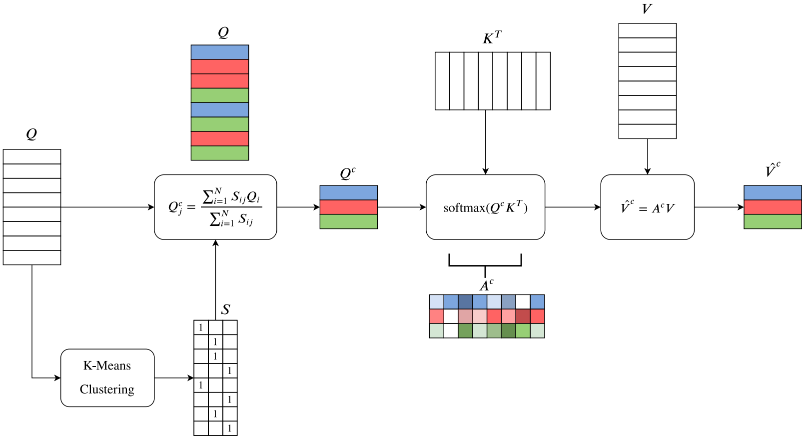
Clustered attention



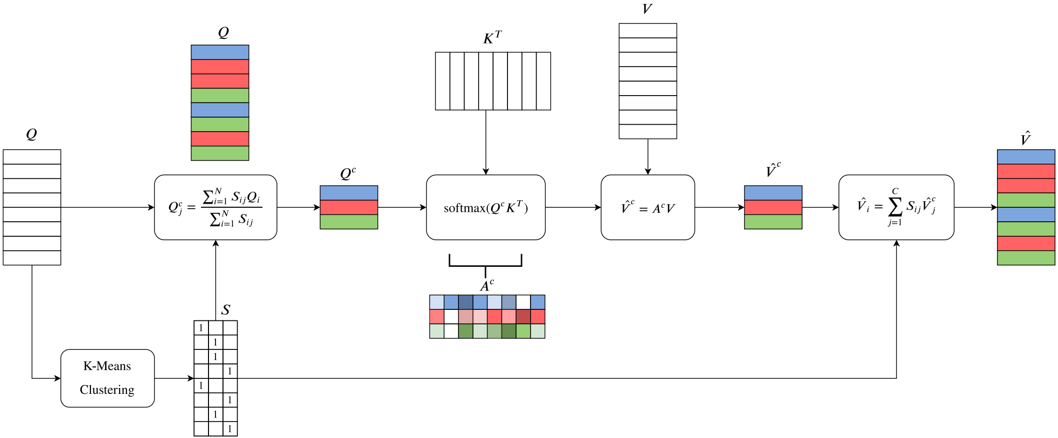
Clustered attention



Clustered attention



Clustered attention



Improved Clustered Attention

For a single **query** Q_i and its **corresponding** cluster **centroid** Q_j^c, standard attention is approximated as:

$$A_i = \text{softmax}\left(Q_i K^T\right) \approx \text{softmax}\left(Q_j^c K^T\right) = A_i^c$$

Improved Clustered Attention

For a single **query** Q_i and its **corresponding** cluster **centroid** Q_j^c, standard attention is approximated as:

$$A_i = \mathrm{softmax}\left(Q_i K^T\right) \approx \mathrm{softmax}\left(Q_j^c K^T\right) = A_i^c$$

Using even **a few exact dot products** improves this approximation.

Improved Clustered Attention

Given a set of key indices $T = \{k_1, k_2, \ldots\}$

$$A^t_{ik} = \begin{cases} w \frac{\exp Q_i K_k^T}{\sum_{r \in T} \exp Q_i K_r^T} & k \in T \\ A^c_{ik} & k \notin T \end{cases}$$

Finally, we show that $|A - A^c|_1 \geq |A - A^t|_1$ which improves our previous approximation.

Experimental setup

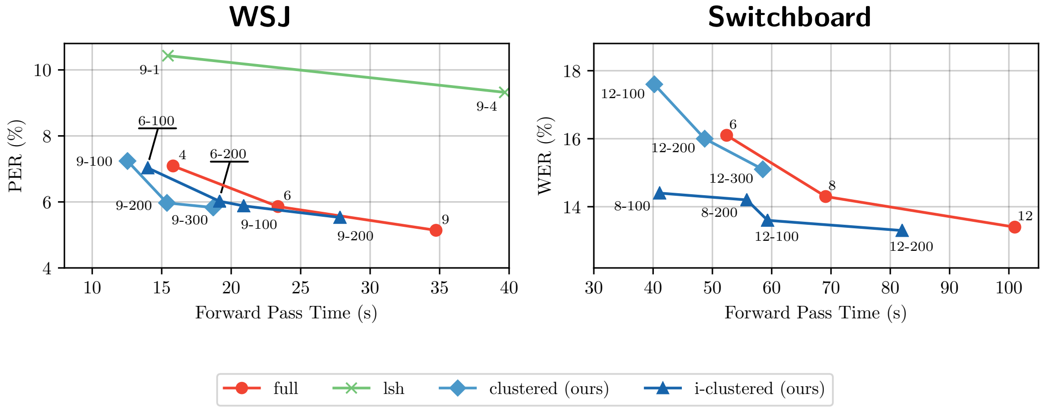
Baselines

- Softmax transformer (Vaswani et al., 2017)
- LSH attention from Reformer (Kitaev et al., 2020)
- FAVOR random Fourier features from Performer (Choromanski et al., 2020)

Experiments

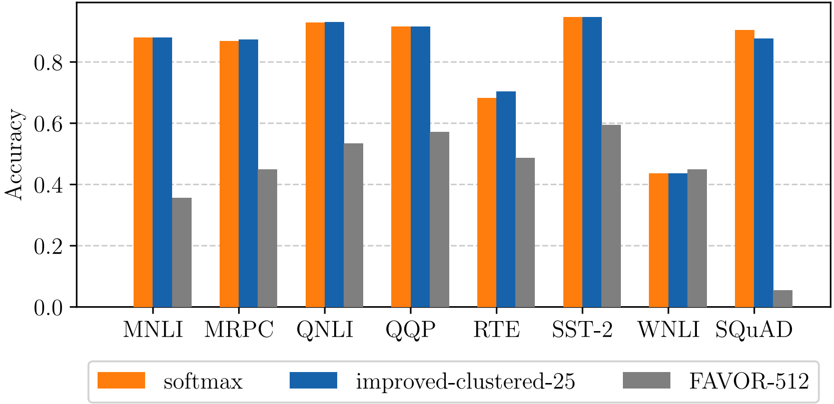
- Automatic speech recognition on WSJ and Switchboard
- Approximation of pretrained RoBERTa on GLUE and SQuAD
- Approximation of pretrained Wav2Vec

Automatic Speech Recognition



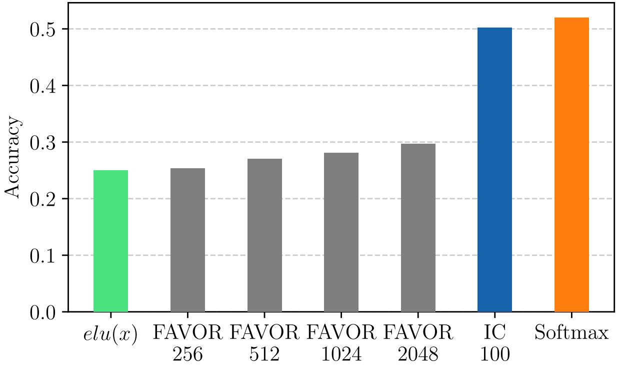
RoBERTa approximation

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



Wav2Vec approximation

Wav2Vec approximation on LibriSpeech.



Take-away messages

Take-away messages

- Transformers are here to stay

Take-away messages

- Transformers are here to stay
- Transformers offer unique opportunities for multi-modal processing

Take-away messages

- Transformers are here to stay
- Transformers offer unique opportunities for multi-modal processing
- Efficient transformers will popularize their use in more modalities

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.

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

References II

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