Tensor Decomposition Techniques for Transformers

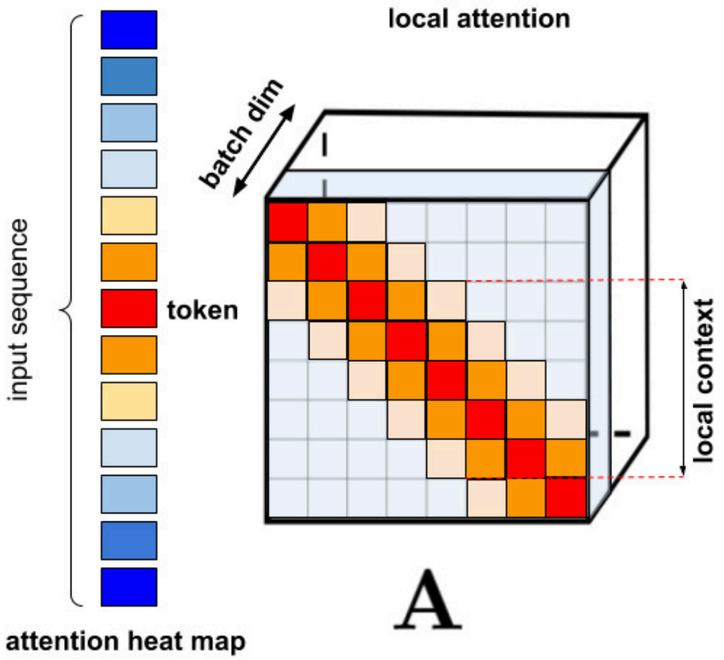
CS618 - Theoretical Foundations of Machine Learning

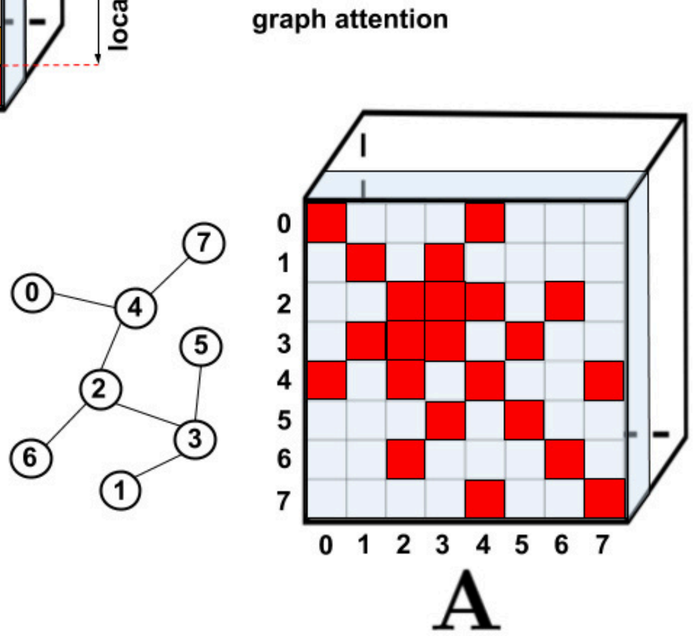
Guntas Singh Saran (22110089)

Hrriday Ruparel (22110099)

Sumeet Sawale (22110234)

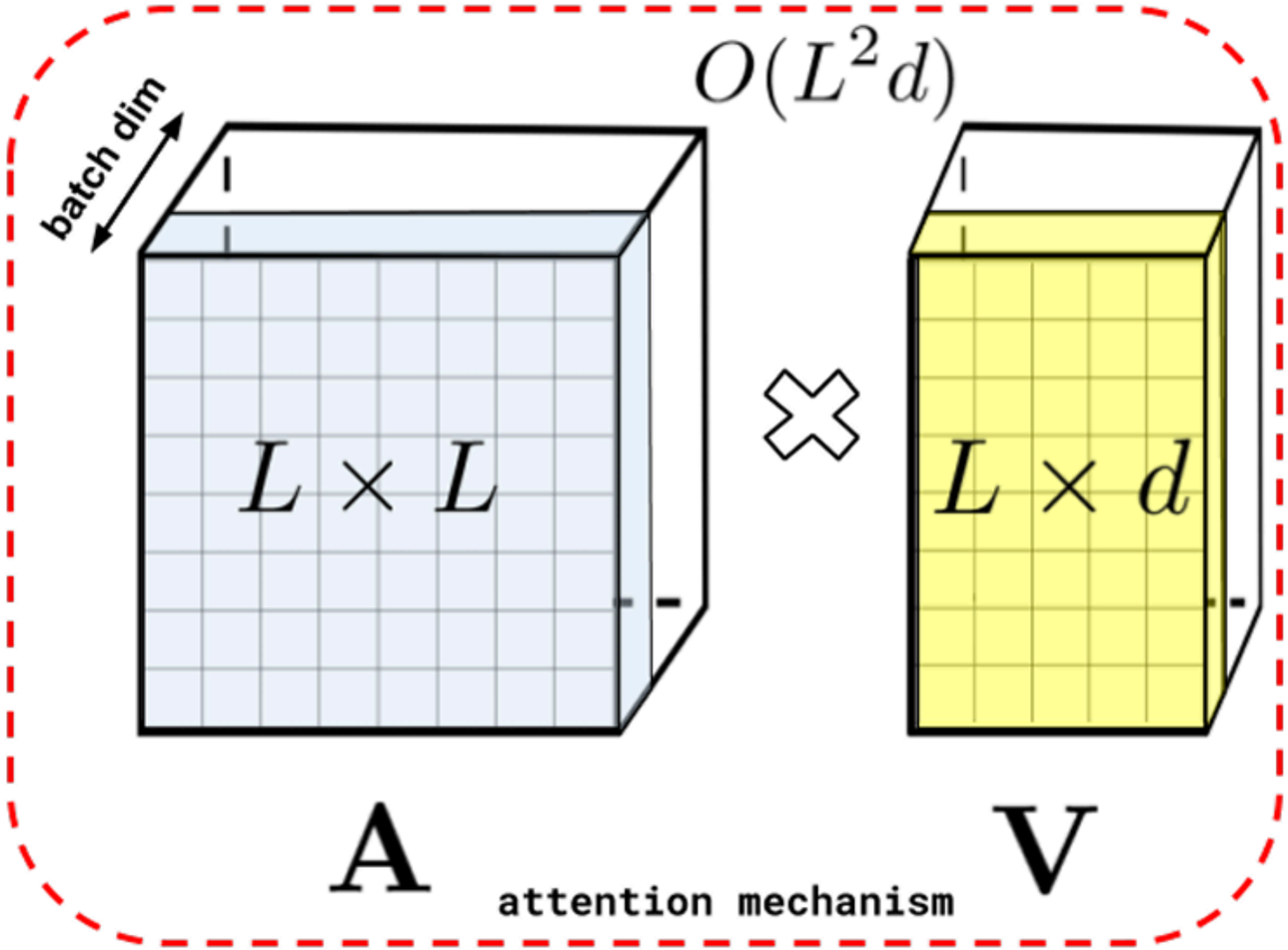
Indian Institue of Technology Gandhinagar
Palaj, Gujarat - 382355





Major Source: https://research.google/blog/rethinking-attention-with-performers/

Why Transformers for Decomposition?



Source: https://arxiv.org/abs/2009.14794

Random Features for Large-Scale Kernel Machines

Rahimi et al. (2007)

The kernel trick is a simple way to generate features for algorithms that depend only on the inner product between pairs of input points. It relies on the observation that any positive definite function $k(\mathbf{x}, \mathbf{y})$ with $\mathbf{x}, \mathbf{y} \in \mathcal{R}^d$ defines an inner product and a lifting ϕ so that the inner product between lifted datapoints can be quickly computed as $\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = k(\mathbf{x}, \mathbf{y})$. The cost of this convenience is that the algorithm accesses the data only through evaluations of $k(\mathbf{x}, \mathbf{y})$, or through the kernel matrix consisting of k applied to all pairs of datapoints. As a result, large training sets incur large computational and storage costs.

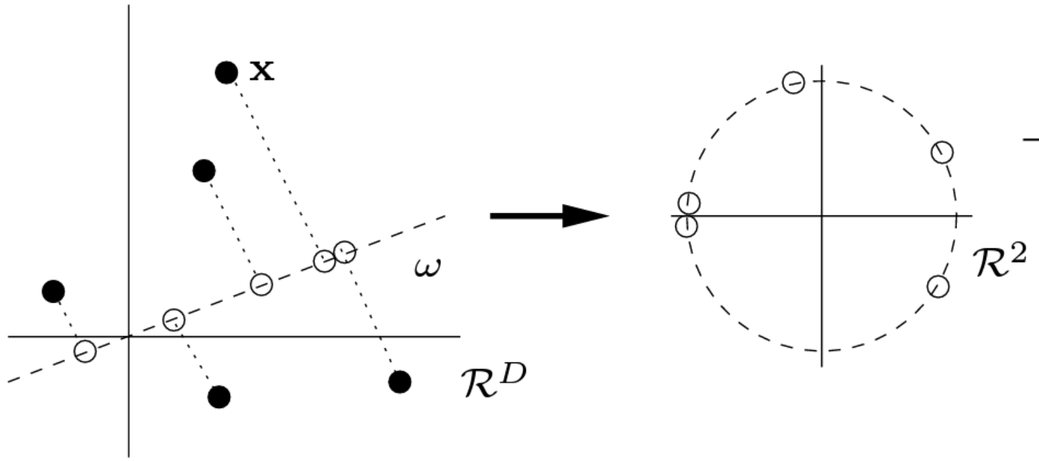
Instead of relying on the implicit lifting provided by the kernel trick, we propose explicitly mapping the data to a low-dimensional Euclidean inner product space using a randomized feature map $\mathbf{z} : \mathcal{R}^d \rightarrow \mathcal{R}^D$ so that the inner product between a pair of transformed points approximates their kernel evaluation:

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \approx \mathbf{z}(\mathbf{x})' \mathbf{z}(\mathbf{y}). \quad (1)$$

Source: https://papers.nips.cc/paper_files/paper/2007/hash/013a006f03dbc5392effeb8f18fda755-Abstract.html

Random Features for Large-Scale Kernel Machines

Rahimi et al. (2007)



Kernel Name	$k(\Delta)$	$p(\omega)$
Gaussian	$e^{-\frac{\|\Delta\|_2^2}{2}}$	$(2\pi)^{-\frac{D}{2}} e^{-\frac{\|\omega\|_2^2}{2}}$
Laplacian	$e^{-\|\Delta\|_1}$	$\prod_d \frac{1}{\pi(1+\omega_d^2)}$
Cauchy	$\prod_d \frac{2}{1+\Delta_d^2}$	$e^{-\|\Delta\|_1}$

Figure 1: Random Fourier Features. Each component of the feature map $\mathbf{z}(\mathbf{x})$ projects $\mathbf{x}$ onto a random direction ω drawn from the Fourier transform $p(\omega)$ of $k(\Delta)$, and wraps this line onto the unit circle in $\mathcal{R}^2$. After transforming two points $\mathbf{x}$ and $\mathbf{y}$ in this way, their inner product is an unbiased estimator of $k(\mathbf{x}, \mathbf{y})$. The table lists some popular shift-invariant kernels and their Fourier transforms. To deal with non-isotropic kernels, the data may be whitened before applying one of these kernels.

Source: https://papers.nips.cc/paper_files/paper/2007/hash/013a006f03dbc5392effeb8f18fda755-Abstract.html

Random Features for Large-Scale Kernel Machines

Rahimi et al. (2007)

$$\Pr\left[|\mathbf{z}(\mathbf{x})'\mathbf{z}(\mathbf{y}) - k(\mathbf{x},\mathbf{y})| \geq \epsilon\right] \leq 2\exp(-D\epsilon^2/2)$$

Algorithm 1 Random Fourier Features.

Require: A positive definite shift-invariant kernel $k(\mathbf{x},\mathbf{y}) = k(\mathbf{x}-\mathbf{y})$.
Ensure: A randomized feature map $\mathbf{z}(\mathbf{x}) : \mathcal{R}^d \rightarrow \mathcal{R}^{2D}$ so that $\mathbf{z}(\mathbf{x})'\mathbf{z}(\mathbf{y}) \approx k(\mathbf{x}-\mathbf{y})$.
Compute the Fourier transform p of the kernel k: $p(\omega) = \frac{1}{2\pi}\int e^{-j\omega'\Delta}k(\Delta)\, d\Delta$.
Draw D iid samples $\omega_1, \cdots, \omega_D \in \mathcal{R}^d$ from p.
Let $\mathbf{z}(\mathbf{x}) \equiv \sqrt{\frac{1}{D}}\left[\cos(\omega_1'\mathbf{x}) \cdots \cos(\omega_D'\mathbf{x}) \sin(\omega_1'\mathbf{x}) \cdots \sin(\omega_D'\mathbf{x})\right]'$.

Source: https://papers.nips.cc/paper_files/paper/2007/hash/013a006f03dbc5392effeb8f18fda755-Abstract.html

Random Feature Attention

Peng et al. (2021)

Attention Sequence Modelling

Let $\{\mathbf{q}_t\}_{t=1}^N$ denote a sequence of N **query** vectors, that attend to sequences of M **key** and **value** vectors. At each timestep, the attention linearly combines the values weighted by the outputs of a softmax:

$$\text{attn}\left(\mathbf{q}_t, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}\right) = \sum_i \frac{\exp\left(\mathbf{q}_t \cdot \mathbf{k}_i / \tau\right)}{\sum_j \exp\left(\mathbf{q}_t \cdot \mathbf{k}_j / \tau\right)} \mathbf{v}_i^\top. \tag{1}$$

τ is the temperature hyperparameter determining how "flat" the softmax is (Hinton et al., 2015).[1]

Calculating attention for a single query takes $\mathcal{O}(M)$ time and space. For the full sequence of N queries the space amounts to $\mathcal{O}(MN)$. When the computation *cannot* be parallelized across the queries, e.g., in autoregressive decoding, the time complexity is quadratic in the sequence length.

Source: https://arxiv.org/abs/2103.02143

Random Feature Attention

Random Feature Methods

Theorem 1 (Rahimi & Recht, 2007). *Let* $\phi : \mathbb{R}^d \to \mathbb{R}^{2D}$ *be a nonlinear transformation:*

$$\phi(\mathbf{x}) = \sqrt{1/D}\left[\sin(\mathbf{w}_1 \cdot \mathbf{x}), \ldots, \sin(\mathbf{w}_D \cdot \mathbf{x}), \cos(\mathbf{w}_1 \cdot \mathbf{x}), \ldots, \cos(\mathbf{w}_D \cdot \mathbf{x})\right]^\top.$$

When d*-dimensional random vectors* $\mathbf{w}_i$ *are independently sampled from* $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_d)$,

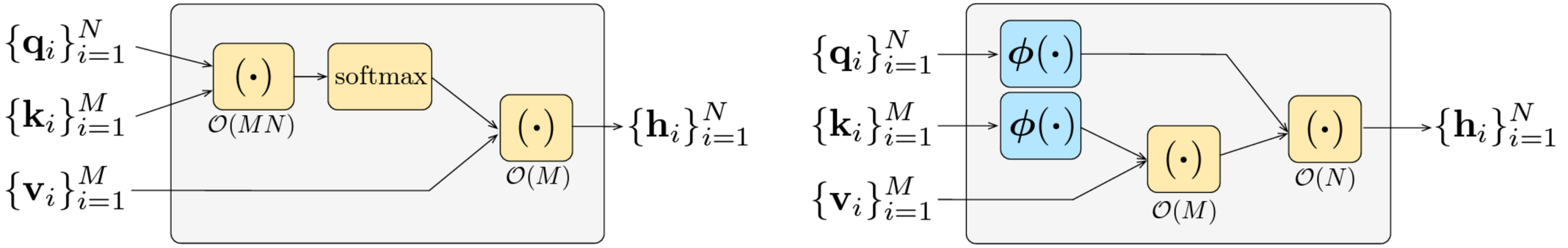
$$\mathbb{E}_{\mathbf{w}_i}\left[\phi(\mathbf{x}) \cdot \phi(\mathbf{y})\right] = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2 / 2\sigma^2\right).$$

Variance of the estimation is inversely proportional to D

Source: https://arxiv.org/abs/2103.02143

Random Feature Attention

Random Feature Methods



(a) Softmax attention.

(b) Random feature attention.

Figure 1: Computation graphs for softmax attention (left) and random feature attention (right). Here, we assume cross attention with source length M and target length N.

Source: https://arxiv.org/abs/2103.02143

Random Feature Attention

Random Feature Methods

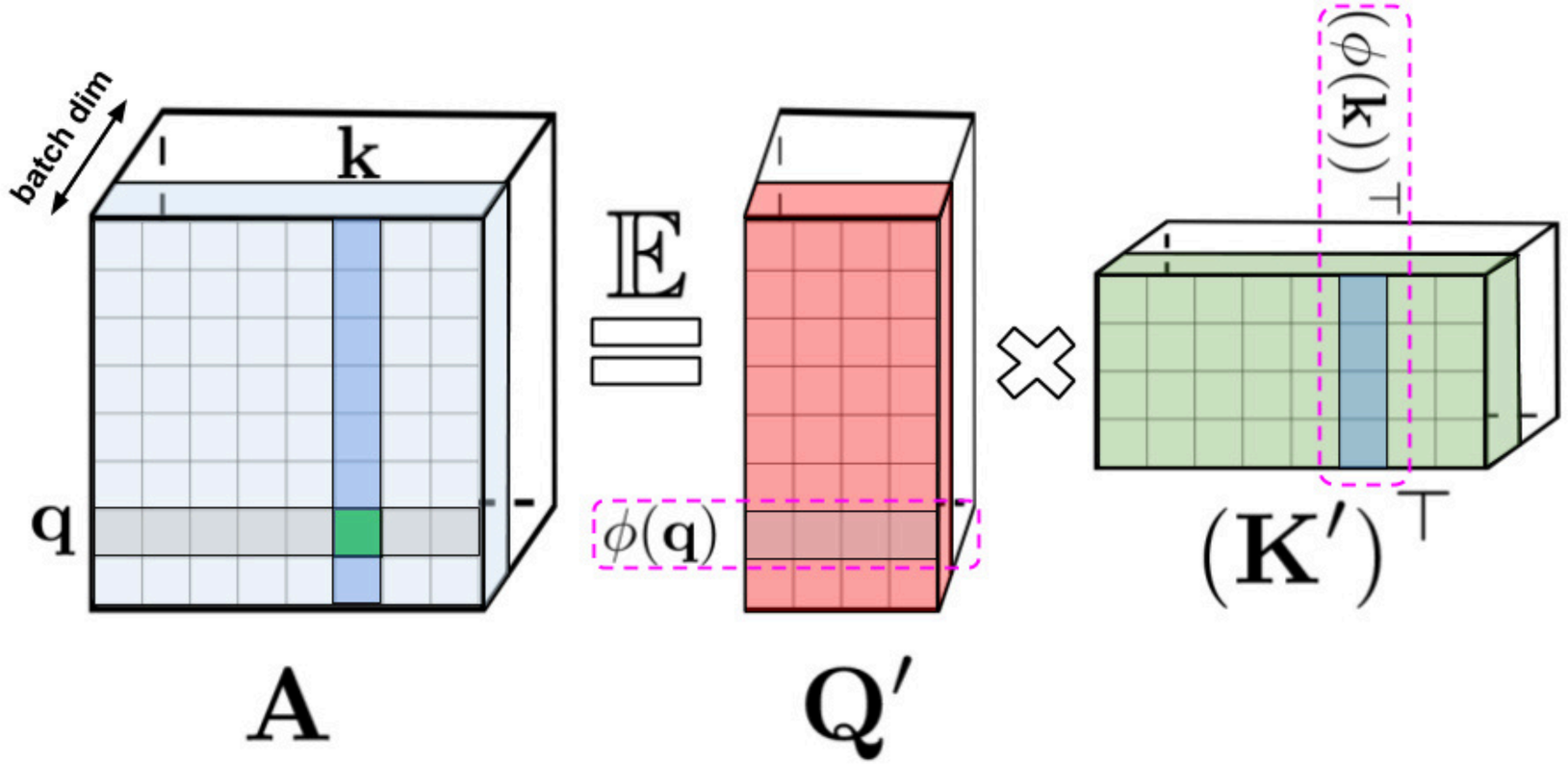
$$\begin{aligned}\exp\left(\mathbf{x}\cdot\mathbf{y}/\sigma^2\right) &= \exp\left(\left\|\mathbf{x}\right\|^2/2\sigma^2 + \left\|\mathbf{y}\right\|^2/2\sigma^2\right)\exp\left(-\left\|\mathbf{x}-\mathbf{y}\right\|^2/2\sigma^2\right)\\ &\approx \exp\left(\left\|\mathbf{x}\right\|^2/2\sigma^2 + \left\|\mathbf{y}\right\|^2/2\sigma^2\right)\ \boldsymbol{\phi}\left(\mathbf{x}\right)\cdot\boldsymbol{\phi}\left(\mathbf{y}\right).\end{aligned}$$

$$\begin{aligned}\mathrm{attn}\left(\mathbf{q}_t,\{\mathbf{k}_i\},\{\mathbf{v}_i\}\right) &= \sum_i \frac{\exp\left(\mathbf{q}_t\cdot\mathbf{k}_i/\sigma^2\right)}{\sum_j \exp\left(\mathbf{q}_t\cdot\mathbf{k}_j/\sigma^2\right)}\mathbf{v}_i^\top\\ &\approx \sum_i \frac{\boldsymbol{\phi}\left(\mathbf{q}_t\right)^\top\boldsymbol{\phi}\left(\mathbf{k}_i\right)\mathbf{v}_i^\top}{\sum_j \boldsymbol{\phi}\left(\mathbf{q}_t\right)\cdot\boldsymbol{\phi}\left(\mathbf{k}_j\right)}\\ &= \frac{\boldsymbol{\phi}\left(\mathbf{q}_t\right)^\top\sum_i \boldsymbol{\phi}\left(\mathbf{k}_i\right)\otimes\mathbf{v}_i}{\boldsymbol{\phi}\left(\mathbf{q}_t\right)\cdot\sum_j \boldsymbol{\phi}\left(\mathbf{k}_j\right)} = \mathrm{RFA}\left(\mathbf{q}_t,\{\mathbf{k}_i\},\{\mathbf{v}_i\}\right).\end{aligned}$$

Source: https://arxiv.org/abs/2103.02143

Random Feature Attention

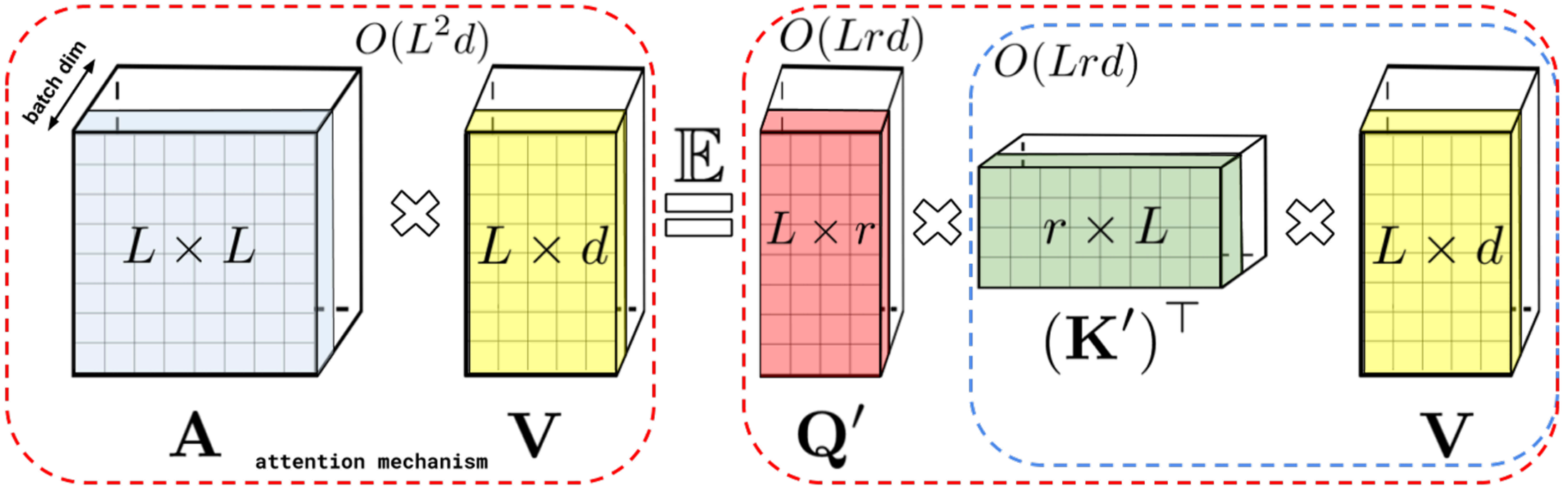
The standard attention matrix can be approximated via lower-rank randomized matrices Q′ and K′ with rows encoding potentially randomized nonlinear functions of the original queries/keys



Source: https://research.google/blog/rethinking-attention-with-performers/

Rethinking Attention with Performers

Choromanski et al. (2022)



Source: https://arxiv.org/abs/2009.14794

RETHINKING ATTENTION WITH PERFORMERS

Krzysztof Choromanski[*1], **Valerii Likhosherstov**[*2], **David Dohan**[*1], **Xingyou Song**[*1]
Andreea Gane[*1], **Tamas Sarlos**[*1], **Peter Hawkins**[*1], **Jared Davis**[*3], **Afroz Mohiuddin**[1]
Lukasz Kaiser[1], **David Belanger**[1], **Lucy Colwell**[1,2], **Adrian Weller**[2,4]
[1]Google [2]University of Cambridge [3]DeepMind [4]Alan Turing Institute

ABSTRACT

We introduce *Performers*, Transformer architectures which can estimate regular (softmax) full-rank-attention Transformers with provable accuracy, but using only linear (as opposed to quadratic) space and time complexity, without relying on any priors such as sparsity or low-rankness. To approximate softmax attention-kernels, Performers use a novel *Fast Attention Via positive Orthogonal Random features* approach (FAVOR+), which may be of independent interest for scalable kernel methods. FAVOR+ can also be used to efficiently model kernelizable attention mechanisms beyond softmax. This representational power is crucial to accurately compare softmax with other kernels for the first time on large-scale tasks, beyond the reach of regular Transformers, and investigate optimal attention-kernels. Performers are linear architectures fully compatible with regular Transformers and with strong theoretical guarantees: unbiased or nearly-unbiased estimation of the attention matrix, uniform convergence and low estimation variance. We tested Performers on a rich set of tasks stretching from pixel-prediction through text models to protein sequence modeling. We demonstrate competitive results with other examined efficient sparse and dense attention methods, showcasing effectiveness of the novel attention-learning paradigm leveraged by Performers.

Source: https://arxiv.org/abs/2009.14794

Rethinking Attention with Performers

Choromanski et al. (2022)

It turns out that by taking ϕ of the following form for functions $f_1, ..., f_l : \mathbb{R} \rightarrow \mathbb{R}$, function $g : \mathbb{R}^d \rightarrow \mathbb{R}$ and deterministic vectors ω_i or $\omega_1, ..., \omega_m \overset{\text{iid}}{\sim} \mathcal{D}$ for some distribution $\mathcal{D} \in \mathcal{P}(\mathbb{R}^d)$:

$$\phi(\mathbf{x}) = \frac{h(\mathbf{x})}{\sqrt{m}}(f_1(\omega_1^\top \mathbf{x}), ..., f_1(\omega_m^\top \mathbf{x}), ..., f_l(\omega_1^\top \mathbf{x}), ..., f_l(\omega_m^\top \mathbf{x})), \tag{5}$$

we can model most kernels used in practice. Furthermore, in most cases $\mathcal{D}$ is isotropic (i.e. with pdf function constant on a sphere), usually Gaussian. For example, by taking $h(\mathbf{x}) = 1, l = 1$ and $\mathcal{D} = \mathcal{N}(0, \mathbf{I}_d)$ we obtain estimators of the so-called PNG-kernels (Choromanski et al., 2017) (e.g. $f_1 = \text{sgn}$ corresponds to the angular kernel). Configurations: $h(\mathbf{x}) = 1, l = 2, f_1 = \sin, f_2 = \cos$ correspond to shift-invariant kernels, in particular $\mathcal{D} = \mathcal{N}(0, \mathbf{I}_d)$ leads to the Gaussian kernel K_{gauss}

Source: https://arxiv.org/abs/2009.14794

Rethinking Attention with Performers

Choromanski et al. (2022)

$$\mathrm{SM}(\mathbf{x}, \mathbf{y}) \stackrel{\mathrm{def}}{=} \exp(\mathbf{x}^\top \mathbf{y}). \tag{6}$$

In the above, without loss of generality, we omit $\sqrt{d}$-renormalization since we can equivalently renormalize input keys and queries. Since: $\mathrm{SM}(\mathbf{x}, \mathbf{y}) = \exp(\frac{\|\mathbf{x}\|^2}{2}) \mathrm{K}_{\mathrm{gauss}}(\mathbf{x}, \mathbf{y}) \exp(\frac{\|\mathbf{y}\|^2}{2})$, based on what we have said, we obtain random feature map unbiased approximation of $\mathrm{SM}(\mathbf{x}, \mathbf{y})$ using trigonometric functions with: $h(\mathbf{x}) = \exp(\frac{\|\mathbf{x}\|^2}{2})$, $l = 2$, $f_1 = \sin$, $f_2 = \cos$. We call it $\widehat{\mathrm{SM}}^{\mathrm{trig}}_m(\mathbf{x}, \mathbf{y})$.

There is however a caveat there. The attention module from (1) constructs for each token, a convex combination of value-vectors with coefficients given as corresponding renormalized kernel scores. That is why kernels producing non-negative scores are used. Applying random feature maps with potentially negative dimension-values ($\sin$ / $\cos$) leads to unstable behaviours, especially when kernel scores close to 0 (which is the case for many entries of $\mathbf{A}$ corresponding to low relevance tokens) are approximated by estimators with large variance in such regions. This results in abnormal behaviours, e.g. negative-diagonal-values renormalizers $\mathbf{D}^{-1}$, and consequently either completely prevents training or leads to sub-optimal models. We demonstrate empirically that this is what happens for $\widehat{\mathrm{SM}}^{\mathrm{trig}}_m$ and provide detailed theoretical explanations showing that the variance of $\widehat{\mathrm{SM}}^{\mathrm{trig}}_m$ is large as approximated values tend to 0 (see: Section 3). This is one of the main reasons why the robust random feature map mechanism for approximating regular softmax attention was never proposed.

Source: https://arxiv.org/abs/2009.14794

Tensor Decomposition Techniques for Attention

The Introduction

Matrix-based factorization techniques for approximating Attention Mechanism have been very well explored.

Tensor-based learning and representation is gaining lot of traction lately

Let’s explore what tensor decomposition has to offer us for approximating Attention Mechanisms

A Tensorized Transformer for Language Modeling

Ma et al. (2019)

A Tensorized Transformer for Language Modeling

Xindian Ma[1], **Peng Zhang**[1*], **Shuai Zhang**[1],
Nan Duan[2], **Yuexian Hou**[1], **Dawei Song**[3], **Ming Zhou**[2]
[1]College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]Microsoft Research Asia, Beijing, China
[3]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
{xindianma, pzhang, szhang96, yxhou}@tju.edu.cn
{nanduan, mingzhou}@microsoft.com
{dwsong}@bit.edu.cn

Abstract

Latest development of neural models has connected the encoder and decoder through a self-attention mechanism. In particular, Transformer, which is solely based on self-attention, has led to breakthroughs in Natural Language Processing (NLP) tasks. However, the multi-head attention mechanism, as a key component of Transformer, limits the effective deployment of the model to a resource-limited setting. In this paper, based on the ideas of tensor decomposition and parameters sharing, we propose a novel self-attention model (namely Multi-linear attention) with Block-Term Tensor Decomposition (BTD). We test and verify the proposed attention method on three language modeling tasks (i.e., PTB, WikiText-103 and One-billion) and a neural machine translation task (i.e., WMT-2016 English-German). Multi-linear attention can not only largely compress the model parameters but also obtain performance improvements, compared with a number of language modeling approaches, such as Transformer, Transformer-XL, and Transformer with tensor train decomposition.

Source: https://arxiv.org/abs/1906.09777

A Tensorized Transformer for Language Modeling

Ma et al. (2019)

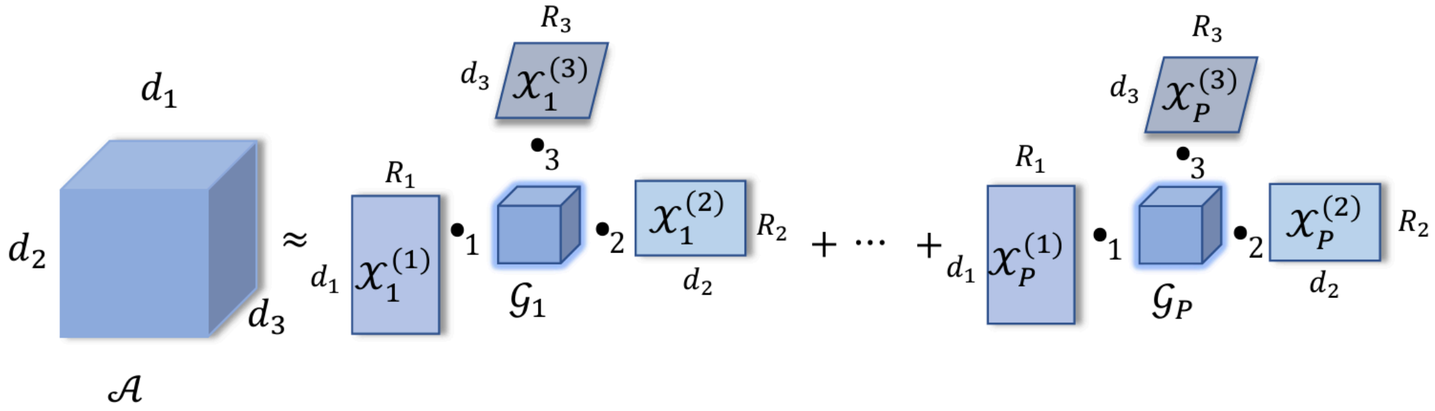


Figure 1: The representation of Block-Term tensor decomposition for a 3-order tensor. $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ is a 3-order tensor, and can be approximated by P Tucker decomposition. P is the CP rank, and R_1, R_2, R_3 are the Tucker rank, respectively. In this paper, we assume that R=R_1=R_2=R_3.

Source: https://arxiv.org/abs/1906.09777

A Tensorized Transformer for Language Modeling

Ma et al. (2019)

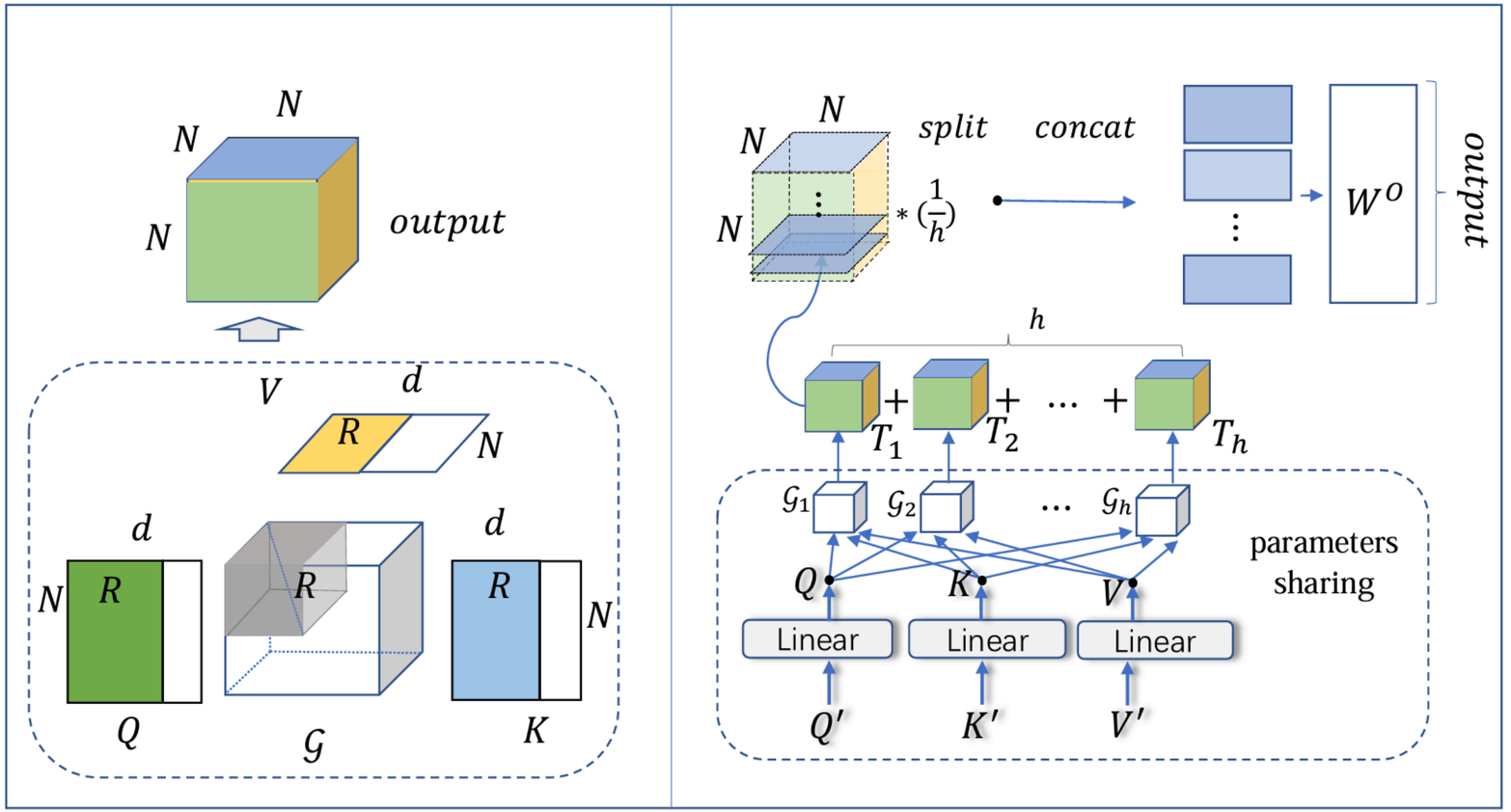


Figure 2: (left) Single-block attention using Tucker decomposition. (right) Multi-linear attention based on Block-Term tensor decomposition.

Source: https://arxiv.org/abs/1906.09777

A Tensorized Transformer for Language Modeling

Ma et al. (2019)

3.1 Single-block Attention by Tucker Decomposition

Before building the Single-block attention, it is necessary to propose the theorem 3.1. The theorem is closely related to attributes of Single-block attention function by Tucker decomposition [33].

Theorem 3.1. *Let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ be basis vectors from the vector space S. Assume that these vectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ are linear independent and Q, K, V can be linearly represented by this set of basis vectors. The output of the attention function in Eq. 2 can be represented by a linear combination of the set of these basis vectors.*

$$Attention(Q, K, V) = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n)M, \tag{4}$$

where $M \in \mathbb{R}^{n \times d}$ is a coefficient matrix, and d is a dimension of these matrices (i.e., Q, K, and V).

Source: https://arxiv.org/abs/1906.09777

A Tensorized Transformer for Language Modeling

Appendix B Theorem 3.1

Let $e_1, ..., e_n$ be basis vectors from the vector space S. Assume that these vectors $e_1, ..., e_n$ are linear independent and Q,K,V can be linearly represented by this set of basis vectors. The output of self-attention function in Eq. 2 (in the paper) can be represented by a linear combination of the set of these basis vectors.

$$Attention\ (\ Q, K, V\ )\ =\ (\ e_1, ..., e_n\ )\ M, \tag{9}$$

where $M \in \mathbb{R}^{n \times d}$ is a coefficient matrix, and d is a dimension of these matrices (i.e., Q, K, and V).

Proof. If Q, K and $V \in$ Span($e_1, ..., e_n$), the linear combination representation of matrices Q,K and V can be written as follows:

$$\begin{cases} Q = (\ e_1, e_2, ..., e_n\ )\ (\ \alpha_1, \alpha_2, ..., \alpha_d\ ) \\ K = (\ e_1, e_2, ..., e_n\ )\ (\ \beta_1, \beta_2, ..., \beta_d\ ) \\ V = (\ e_1, e_2, ..., e_n\ )\ (\ \xi_1, \xi_2, ..., \xi_d\ ) \end{cases} \tag{10}$$

The self-attention function is written as follows [37]:

$$Attention\ (\ Q, K, V\ )\ = softmax\ (\ \frac{QK^T}{\sqrt{d}}\ )\ V, \tag{11}$$

where QK^T can be computed as follows:

Source: https://ar5iv.labs.arxiv.org/html/1906.09777

A Tensorized Transformer for Language Modeling

where QK^T can be computed as follows:

$$QK^T = (e_1, e_2, ..., e_n)(\alpha_1, \alpha_2, ..., \alpha_d)(\beta_1, \beta_2, ..., \beta_d)^T (e_1, e_2, ..., e_n)^T \tag{12}$$

As a result, the input of $softmax$ function is a product of coefficient matrices $(\alpha_1, ..., \alpha_d)$ and $(\beta_1, ..., \beta_d)^T$. Then, we have

$$softmax\left(\frac{QK^T}{\sqrt{d}}\right) = (e_1, ..., e_n)\, softmax(A / \sqrt{d})\, (e_1, ..., e_n)^T \tag{13}$$

where the matrix A is equal to $(\alpha_1, ..., \alpha_d)(\beta_1, ..., \beta_d)^T$. Therefore, the attention representation can be written as follows:

$$\begin{aligned} softmax\left(\frac{QK^T}{\sqrt{d}}\right) V &= (e_1, e_2, ..., e_n)\, softmax(A / \sqrt{d})\, (\xi_1, \xi_2, ..., \xi_d) \\ &= (e_1, e_2, ..., e_n)\, M \end{aligned} \tag{14}$$

where the matrix M is equal to $softmax(A / \sqrt{d})(\xi_1, \xi_2, ..., \xi_d)$. The $softmax(A / \sqrt{d})$ is to normalize the coefficient matrices of Q and K. It turns out that the output of the attention function [37] can be represented by a linear combination of the set of basic vectors. ∎

Source: https://ar5iv.labs.arxiv.org/html/1906.09777

Testing their Claim

```
def custom_softmax(A, B, d):
    return np.exp(A @ B.T / np.sqrt(d)) / np.exp(A @ B.T / np.sqrt(d)).sum(axis=1)[:, None]

d, n = 64, 1000

E = np.random.randn(n, n)
Q, R = np.linalg.qr(E)

print(np.allclose(Q @ Q.T, np.eye(n)), np.allclose(Q.T @ Q, np.eye(n)))
print()

alpha = np.random.randn(n, d)
beta = np.random.randn(n, d)
gamma = np.random.randn(n, d)

Queries = Q @ alpha
Keys = Q @ beta
Values = Q @ gamma

softmax = custom_softmax(Queries, Keys, d)
```

Testing their Claim

```
print("QK^T")
print(np.dot(Queries, Keys.T) * np.sqrt(1/d))
print()

print("Softmax")
```

```
(base)  guntas13  Desktop  python3 test.py
True True

QK^T
[[-0.26528765  1.67371306 -1.34290129 ...  -0.31680232 -0.50982631
  -0.59574455]
 [ 0.91013638 -0.37547918 -1.0948681  ...   1.2673479   0.99007252
   1.26981028]
 [-0.88614298 -0.11779822  0.07055755 ...   0.45009419 -0.01880774
   0.07568967]
 ...
 [-0.40088556 -0.46809031 -0.9511074  ...   0.6822173   0.57194242
   0.69742594]
 [ 0.87864523  0.96000659 -1.04082743 ...   1.21353815  1.0388592
  -0.91422576]
 [ 0.73556473 -0.82981124 -0.34632993 ...  -1.03546258 -0.20403345
   1.45279406]]

Softmax
[[0.00053433 0.00371455 0.00018189 ... 0.0005075  0.00041841 0.00038397]
 [0.00162831 0.0004502  0.00021927 ... 0.00232741 0.00176382 0.00233315]
 [0.00019791 0.00042673 0.00051518 ... 0.00075298 0.00047113 0.00051783]
 ...
 [0.00039023 0.00036487 0.00022509 ... 0.00115268 0.00103233 0.00117034]
 [0.00157445 0.00170791 0.00023095 ... 0.00220076 0.00184803 0.00026212]
 [0.00116363 0.00024321 0.00039441 ... 0.000198   0.00045473 0.00238398]]
```

Testing their Claim

```python
softmax_alpha_beta = custom_softmax(Q @ alpha, Q @ beta, d)
print("QK^T = E @ Alpha @ Beta^T @ E^T")
print(softmax_alpha_beta)
print()

softmax_proposed = custom_softmax(alpha, beta, d)
softmax_proposed = Q @ softmax_proposed @ Q.T

print("E @ Softmax(Alpha, Beta) @ E^T")
print(softmax_proposed)
print()

res = np.allclose(softmax_alpha_beta, softmax_proposed)
print(f"Are the two softmaxes equal? {res}")
```

```
QK^T = E @ Alpha @ Beta^T @ E^T
[[0.00053433 0.00371455 0.00018189 ... 0.0005075  0.00041841 0.00038397]
 [0.00162831 0.0004502  0.00021927 ... 0.00232741 0.00176382 0.00233315]
 [0.00019791 0.00042673 0.00051518 ... 0.00075298 0.00047113 0.00051783]
 ...
 [0.00039023 0.00036487 0.00022509 ... 0.00115268 0.00103233 0.00117034]
 [0.00157445 0.00170791 0.00023095 ... 0.00220076 0.00184803 0.00026212]
 [0.00116363 0.00024321 0.00039441 ... 0.000198   0.00045473 0.00238398]]

E @ Softmax(Alpha, Beta) @ E^T
[[ 5.20356242e-05  3.19749277e-03 -1.43021237e-03 ... -8.47230415e-04
  -6.09793412e-04 -9.13039167e-04]
 [ 3.02886157e-03 -1.61912433e-04 -1.32203066e-03 ...  2.30419424e-04
   1.68983005e-03  8.95218684e-04]
 [-2.08068961e-03  1.30593463e-03  9.45153032e-05 ...  2.60531288e-04
   9.76042803e-04  4.92785925e-04]
 ...
 [-7.04271015e-04 -7.36926664e-04 -2.21523128e-03 ...  5.33885030e-04
   4.01152900e-04 -2.66883020e-04]
 [ 8.72478218e-04  4.76256678e-04 -1.82455050e-03 ...  7.48637188e-04
   4.62971304e-03 -9.30215542e-04]
 [ 1.12883271e-05 -1.75998128e-03 -1.80975077e-04 ... -3.07868752e-03
  -7.28096083e-04  1.61705421e-03]]

Are the two softmaxes equal? False
```

The Flaw

$$\mathrm{softmax}(UAU^T) \neq U\,\mathrm{softmax}(A)\,U^T$$

$$softmax\left(\frac{QK^T}{\sqrt{d}}\right) \neq (e_1, ..., e_n)\, softmax\left(A / \sqrt{d}\right) (e_1, ..., e_n)^T$$

Source: https://ar5iv.labs.arxiv.org/html/1906.09777

A Tensorized Transformer for Language Modeling

Under the same conditions as in Theorem 3.1 and the value of N is equal to the value of d, the Single-block attention representation Eq. 5 (in the paper) can reconstruct the *Scaled Dot-Product attention* in Eq. 2 (in the paper) by the summing over the tensor (i.e., the output of Single-block attention function) according to the second index. It holds that:

$$Attention\left(Q,K,V\right)_{i,m}=\sum_{j=1}^{N}Atten_{TD}\left(\mathcal{G};Q,K,V\right)_{i,j,m}, \tag{15}$$

where i, j and m are the indices of the Single-block attention output (i.e., a 3-order tensor). $Atten_{TD}\left(\cdot\right)$ is the function of the Single-block attention based on Tucker decomposition. i and m are the indices of outputs (i.e., a matrix) from Eq. 2 (in the paper).

Source: https://ar5iv.labs.arxiv.org/html/1906.09777

A Tensorized Transformer for Language Modeling

Proof. In Theorem 3.1, we have proved the results about the attention function can be represented by a linear combination of basis vectors. Therefore, we can represent the self-attention function in Eq. 2 (in the paper) by the form as follows:

$$Attention\left(Q,K,V\right) = \Theta QK^{T}V \tag{16}$$

where Θ is a normalization factor matrix, which can be used to replace the use of a $sofmax$ function. We assume that Θ contains all the non-zero elements of the core tensor $\mathcal{G}$. The self-attention in Eq. 2 (in the paper) can be re-written as follows:

$$X_{i,m} = \sum_{k=1}^{N}\sum_{r=1}^{R}\Theta_{i,m}Q_{i,r}K_{k,r}V_{k,m} \tag{17}$$

where N is the length of a sentence, $X_{i,m} = Attention\left(Q,K,V\right)_{i,m}$ is the entry of the output from the self-attention, and R is equal to d. Here the core tensor $\mathcal{G}$ is same as that in Eq. 7 (in the paper). Then, the Single-block attention (a 3-order tensor) can be represented as follows:

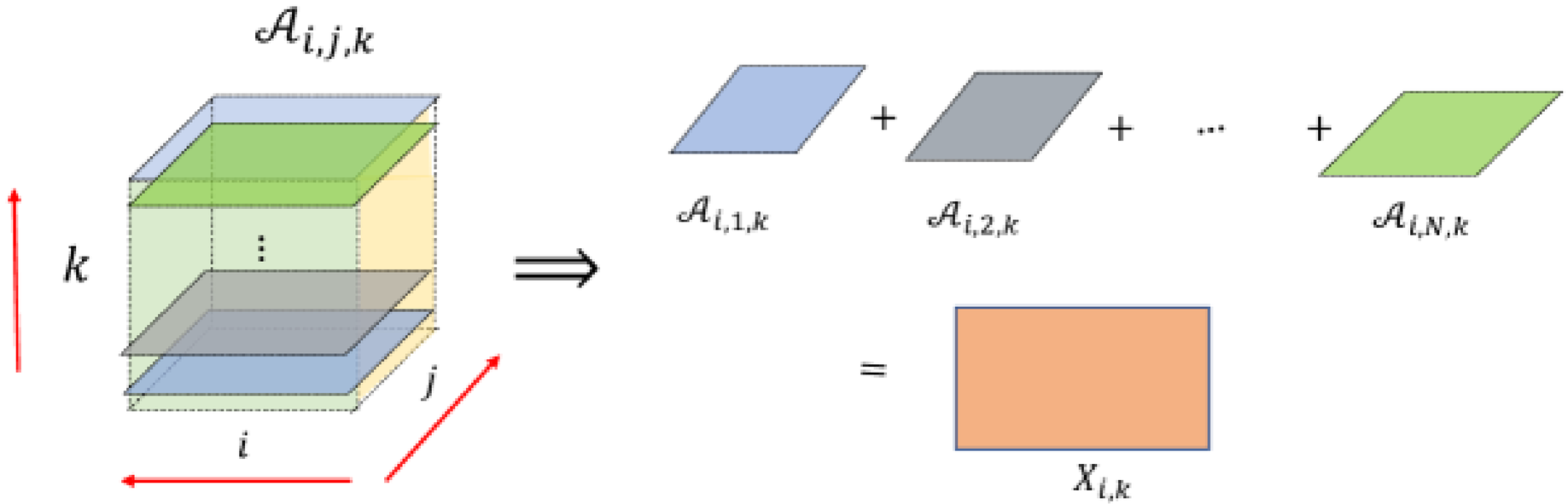
$$\mathcal{A}_{i,j,m} = \sum_{p}^{R}\sum_{q}^{R}\sum_{r}^{R}\mathcal{G}_{p,q,r}Q_{i,p}K_{j,p}V_{m,r} \tag{18}$$

where $\mathcal{A}$ is a 3-order tensor, which is equal to $Atten_{TD}\left(\mathcal{G};Q,K,V\right)$. Accordingly, $\mathcal{A}_{i,j,m}$ is a entry in tensor $\mathcal{A}$ and is equal to $Attention_{TDi,j,m}$ in Eq. 15. Next, we aim to prove Eq. 7 can be established. Therefore, we need to establish the relation between Eq. 18 and Eq. 17. Since the core tensor $\mathcal{G}$ is a special tensor (i.e., diagonal tensor), Eq. 18 can be written as follows:

$$\mathcal{A}_{i,j,m} = \sum_{r=1}^{R}\mathcal{G}_{r,r,r}Q_{i,r}K_{j,r}V_{m,r} \tag{19}$$

Source: https://ar5iv.labs.arxiv.org/html/1906.09777

A Tensorized Transformer for Language Modeling



$$X_{i,m} = \sum_{k=1}^{N} \sum_{r=1}^{R} \Theta_{i,m} Q_{i,r} K_{k,r} V_{k,m} \qquad X_{i,m} = \sum_{r=1}^{R} \sum_{j=1}^{N} \mathcal{G}_{rrr} Q_{i,r} K_{j,r} V_{m,r}$$

Source: https://ar5iv.labs.arxiv.org/html/1906.09777

THANK YOU

Guntas Singh Saran (guntassingh.saran@iitgn.ac.in)

Hrriday V. Ruparel (hrriday.ruparel@iitgn.ac.in)

Sumeet Sawale (sumeet.sawale@iitgn.ac.in)

Indian Institue of Technology Gandhinagar
Palaj, Gujarat - 382355

Rethinking Attention with Performers

Choromanski et al. (2022)

Lemma 1 (Positive Random Features (PRFs) for Softmax)**.** *For* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, $\mathbf{z} = \mathbf{x} + \mathbf{y}$ *we have:*

$$\mathrm{SM}(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{\omega \sim \mathcal{N}(0, \mathbf{I}_d)} \left[\exp\left(\omega^\top \mathbf{x} - \frac{\|\mathbf{x}\|^2}{2}\right) \exp\left(\omega^\top \mathbf{y} - \frac{\|\mathbf{y}\|^2}{2}\right) \right] = \Lambda \mathbb{E}_{\omega \sim \mathcal{N}(0, \mathbf{I}_d)} \cosh(\omega^\top \mathbf{z}), \quad (7)$$

where $\Lambda = \exp(-\frac{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2}{2})$ *and* cosh *is hyperbolic cosine. Consequently, softmax-kernel admits a positive random feature map unbiased approximation with* $h(\mathbf{x}) = \exp(-\frac{\|\mathbf{x}\|^2}{2})$, $l = 1$, $f_1 = \exp$ *and* $\mathcal{D} = \mathcal{N}(0, \mathbf{I}_d)$ *or:* $h(\mathbf{x}) = \frac{1}{\sqrt{2}} \exp(-\frac{\|\mathbf{x}\|^2}{2})$, $l = 2$, $f_1(u) = \exp(u)$, $f_2(u) = \exp(-u)$ *and the same* $\mathcal{D}$ *(the latter for further variance reduction). We call related estimators:* $\widehat{\mathrm{SM}}^{+}_{m}$ *and* $\widehat{\mathrm{SM}}^{\mathrm{hyp}+}_{m}$.

Theorem 4 (uniform convergence for attention approximation)**.** *Assume that* L_2*-norms of queries/keys are upper-bounded by* $R > 0$. *Define* $l = Rd^{-\frac{1}{4}}$ *and take* $h^* = \exp(\frac{l^2}{2})$. *Then for any* $\epsilon > 0$, $\delta = \frac{\epsilon}{(h^*)^2}$ *and the number of random projections* $m = \Theta(\frac{d}{\delta^2} \log(\frac{4d^{\frac{3}{4}} R}{\delta}))$ *the following holds for the attention approximation mechanism leveraging estimators* $\widehat{\mathrm{SM}}^{\mathrm{trig}}$ *with ORFs:* $\|\widehat{\mathbf{A}} - \mathbf{A}\|_\infty \leq \epsilon$ *with any constant probability, where* $\widehat{\mathbf{A}}$ *approximates the attention matrix* $\mathbf{A}$.

Source: https://arxiv.org/abs/2009.14794

Tucker Decomposition