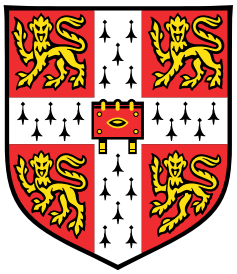
Random Features for Efficient Attention Approximation



Valerii Likhosherstov

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Sidney Sussex College

December 2022

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. I further state that no substantial part of my thesis has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Valerii Likhosherstov
December 2022

Random Features for Efficient Attention Approximation

Valerii Likhosherstov

Transformers are, perhaps, the most widespread architectures in today's landscape of deep learning. They, however, do not scale well with long sequences, resulting in $O(L^2)$ computational complexity for the sequence length L. In this thesis, we propose a holistic approach for reducing this complexity to linear $O(L)$ via an unbiased approximation of the softmax kernel appearing in self-attention, the main component in the Transformer backbone. The obtained efficient Transformer architecture is referred to as Performer. Compared to other developments in the area of efficient Transformers, Performer is theory-grounded and the problem of long-sequence processing can be reduced to a theoretical derivation of random features with certain properties minimizing the variance of the approximation. This thesis describes an evolution of mechanisms for random-feature approximation: from the so-called FAVOR, to FAVOR+ and, finally, to FAVOR++. The FAVOR++ mechanism has the tightest concentration properties and the best performance in practice. On the way to FAVOR++, we also discuss several extensions of the proposed efficient self-attention mechanism, among which are masked self-attention, sub-linear memory Performers, generalized attention, and more. For each proposed method, this thesis contains empirical evaluations in real-life large-scale learning setups and thorough theoretical analyses with proofs.

Acknowledgements

Firstly, I would like to thank my supervisor Adrian Weller and my unofficial co-supervisor Krzysztof Choromanski for their huge support throughout the whole duration of my PhD, both in terms of research and life situations of any kind.

Secondly, I would like to acknowledge people I had a chance of working with: David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, Avinava Dubey, Frederick Liu, Deepali Jain, Michael S Ryoo, Jake Varley, Andy Zeng, Vikas Sindhwani, Anurag Arnab, Mario Lucic, Yi Tay, Tom Weingarten, Wojciech Gajewski, and, the last but not the least, Mostafa Dehghani.

I would also like to acknowledge my advisor José Miguel Hernández-Lobato, other professors at CBL (Zoubin Ghahramani, Carl Rasmussen, Rich Turner), CBL administrator Rachel Fogg and my lab mates. I would like to further acknowledge members of my PhD degree committee: Rich Turner and Zoltán Szabó.

My PhD would've been impossible without people who supported me during the earlier stages of my research path: Pavel Braslavski and Mikhail Khrushchev who supervised me during my undergrad, Misha Chertkov and Yury Maximov (my Master's degree supervisor and co-supervisor respectively). My love for STEM was nurtured by my high school teachers: Sofya Pavlovna Khomenko (math), Aleksey Vadimovich Ivanov (physics), Olga Viktorovna Inisheva (physics), and Svetlana Leonidovna Sandakova (programming).

Lastly, I'm thankful to my partner, my friends Ronald and Vasilii, my mother, and my grandfather.

Table of contents

List of figures

List of tables

Nomenclature

Unbolded z represents a scalar (real or complex number), boldface $\mathbf{z}$ represents a vector, and a capital boldface $\mathbf{Z}$ represents a matrix or tensor of a higher dimension, with an exception of a Gaussian kernel matrix denoted as $\mathscr{K}$. The reason for this is to distinguish the kernel matrix with a "key" matrix notation $\mathbf{K}$ which is standard in the Transformer literature.

Mathematical notation

Symbol	Description
$\mathbb{N}$	a set of *positive* integers
$\mathbf{z}_l$, $\mathbf{z} \in \mathbb{C}^d$	l'th element of the vector $\mathbf{z}$
$\mathbf{Z}_{i,j}$, $\mathbf{Z} \in \mathbb{C}^{d_1 \times d_2}$	(i,j)'th element of the matrix $\mathbf{Z}$
$\mathbf{Z}_i$, $\mathbf{Z} \in \mathbb{C}^{d_1 \times \cdots \times d_t}$	i'th slice of tensor $\mathbf{Z}$ along its first dimension (tensor of shape $d_2 \times \cdots \times d_t$)
$\mathbf{Z}_{:,j}$, $\mathbf{Z} \in \mathbb{C}^{d_1 \times d_2}$	j'th column of matrix $\mathbf{Z}$ (vector of length d_2)
$\mathbf{Z}_{:j}$, $\mathbf{Z} \in \mathbb{C}^{d_1 \times \cdots \times d_t}$	subtensor of shape $j \times d_2 \times \cdots \times d_t$ containing i first slices of the tensor $\mathbf{Z}$ along the first dimension
$\mathbf{Z}_{i:j}$, $\mathbf{Z} \in \mathbb{C}^{d_1 \times \cdots \times d_t}$	subtensor of shape $(j-i+1) \times d_2 \times \cdots \times d_t$ containing slices from $i \geq 1$ to $j > i$ of the tensor $\mathbf{Z}$ along the first dimension
$\|\mathbf{z}\|$, $\mathbf{z} \in \mathbb{R}^d$	L_2-norm $\sqrt{\sum_{l=1}^d \mathbf{z}_l^2}$
$\|\mathbf{Z}\|_\infty$, $\mathbf{Z} \in \mathbb{R}^{d_1 \times \cdots \times d_t}$	L_∞-norm of tensor $\mathbf{Z}$
$K(\mathbf{x},\mathbf{y})$, $\mathbf{x},\mathbf{y} \in \mathbb{R}^d$	Gaussian kernel $\exp\left(-\frac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2\right)$
$\mathscr{K}$	Gaussian kernel matrix computed for $\{\mathbf{x}^{(i)} \in \mathbb{R}^d\}_{1 \leq i \leq L}$ and $\{\mathbf{y}^{(j)} \in \mathbb{R}^d\}_{1 \leq j \leq L'}$, namely $\mathscr{K} = (K(\mathbf{x}^{(i)}, \mathbf{y}^{(j)}))_{i,j=1}^{L,L'} \in \mathbb{R}^{L \times L'}$
$\mathbf{I}_d$	identity matrix of size $d \times d$
$\mathbf{1}_{d_1 \times \cdots \times d_t}$	tensor of size $d_1 \times \cdots \times d_t$ with all ones
$\mathbf{0}_{d_1 \times \cdots \times d_t}$	tensor of size $d_1 \times \cdots \times d_t$ with all zeros
$\mathrm{diag}(\mathbf{z})$, $\mathbf{z} \in \mathbb{C}^d$	diagonal matrix of size $d \times d$ with $\mathbf{z}$ on the diagonal
$\mathrm{tril}(\mathbf{Z})$, $\mathbf{Z} \in \mathbb{C}^{d \times d}$	copy of $\mathbf{Z}$ but with entries above the main diagonal zeroed out
$\mathrm{Tr}(\mathbf{Z})$, $\mathbf{Z} \in \mathbb{C}^{d \times d}$	trace of the square matrix $\mathbf{Z}$, i.e. the sum of its diagonal entries
$\mathrm{PS}(\mathbf{Z})$, $\mathbf{Z} \in \mathbb{C}^{d_1 \times \cdots \times d_t}$	prefix sum along the first dimension of $\mathbf{Z}$, i.e. $(\sum_{j=1}^i \mathbf{Z}_j)_{i=1}^{d_1} \in \mathbb{C}^{d_1 \times \cdots \times d_t}$

Symbol	Description
$\mathbf{Z}^{(1)} \circ \mathbf{Z}^{(2)}$, $\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)} \in \mathbb{C}^{d_1 \times d_2}$	Hadamard product of matrices $\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(2)}$, $(\mathbf{Z}^{(1)}\mathbf{Z}^{(2)})_{i,j=1}^{d_1,d_2} \in \mathbb{C}^{d_1 \times d_2}$
$\rho[\mathbf{Z}]$, $\mathbf{Z} \in \mathbb{C}^{d_1 \times \cdots \times d_t}$	elementwise application of a unary function ρ to a tensor $\mathbf{Z}$, a tensor of shape $d_1 \times \cdots \times d_t$
$[\mathbf{Z}]^d$, $\mathbf{Z} \in \mathbb{C}^{d_1 \times \cdots \times d_t}$	raising each element of tensor $\mathbf{Z}$ to the power d, a tensor of shape $d_1 \times \cdots \times d_t$
$\mathrm{sign}(z)$, $z \in \mathbb{R}$	-1 if $z < 0$, 0 if $z = 0$ and 1 if $z > 0$
$\mathrm{erf}(z)$, $z \in \mathbb{R}$	error function $\frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} \mathrm{d}t$
$\mathrm{ReLU}(z)$, $z \in \mathbb{R}$	rectified linear unit, z if $z \geq 0$ and 0 otherwise
$\mathrm{ELU}(z)$, $z \in \mathbb{R}$	exponential linear unit, z if $z \geq 0$ and $\exp(z) - 1$ otherwise
$\mathrm{GeLU}(z)$, $z \in \mathbb{R}$	Gaussian linear unit, $\frac{1}{2} z \left(1 + \mathrm{erf}\left(z/\sqrt{2}\right)\right)$
i	imaginary unit
$\mathrm{Re}\,(z)$, $z \in \mathbb{C}$	real part of z
$\mathrm{Im}\,(z)$, $z \in \mathbb{C}$	imaginary part of z
$\lvert z \rvert$, $z \in \mathbb{C}$	magnitude of z
$\sqrt[d]{z}$, $z \in \mathbb{C}$	principal d'th root of a complex number, i.e. the d'th root with the biggest real part, and, if there are two roots like that, the one with the positive imaginary part. Also, we denote $\sqrt{z} = \sqrt[2]{z}$
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with the mean μ and variance σ^2
$\chi(d)$	χ-distribution with d degrees of freedom
$\lceil z \rceil$, $z \in \mathbb{R}$	the smallest integer number not smaller than x
$\lfloor z \rfloor$, $z \in \mathbb{R}$	the biggest integer number not bigger than x
$\binom{k}{k_1,\ldots,k_M}$	multinomial coefficient $\frac{k!}{k_1!\ldots k_M!}$ where $k = k_1 + \cdots + k_M$
$p_{\mathrm{SG}}(\boldsymbol{\omega})$, $\boldsymbol{\omega} \in \mathbb{R}^d$	$(2\pi)^{-d/2} \exp\left(-\frac{1}{2}\lVert\boldsymbol{\omega}\rVert^2\right)$, probability density function of the standard multivariate Gaussian distribution
$\mathbb{S}^{d-1}$	sphere $\{\mathbf{x} \in \mathbb{R}^d \,\vert\, \lVert\mathbf{x}\rVert = 1\}$
$\mathrm{Unif}(\mathbb{S}^{d-1})$	uniform distribution on $\mathbb{S}^{d-1}$
$\mathbb{R}_+$	set $\{x \in \mathbb{R} \,\vert\, x > 0\}$
$\mathscr{M}_Z(\phi)$	moment generating function $\mathbb{E}(\exp(\phi Z))$ of the random variable Z, $\phi \in \mathbb{R}$

Abbreviations

Abbreviation	Description
RFs	random features
TrigRFs	trigonometric random features
ORFs	orthogonal random features
FAVOR	*F*ast *A*ttention *V*ia *O*rthogonal *R*andom features
GA	generalized attention
PosRFs	positive random features
FAVOR+	*F*ast *A*ttention *V*ia *positive O*rthogonal *R*andom features

Abbreviation	Description
SLiM Performer	sub-linear memory Performer
GPU	graphics processing unit
TPU	tensor processing unit
FLOP	floating-point operation
LSH	locality-sensitive hashing
GERFs	generalized exponential random features
DIRFs	discretely-induced random features
OPRFs	optimal positive random features
PoisRFs	Poisson random features
GeomRFs	geometric random features
CRTs	chef's random tables

Chapter 1

Introduction

1.1 Motivation for thesis

Proposed in 2017, Transformer networks established a monopoly among state-of-the-art deep learning solutions. With the ability to scale well as the amount of training data and the number of parameters increases, Transformers beat other methods in natural language processing tasks (machine translation, dialog systems, named entity recognition, speech recognition, text summarization, question answering, sentiment analysis, etc.), computer vision tasks (image, video and audio recognition, segmentation, super-resolution, visual question answering, video understanding, point cloud classification and segmentation, text-to-image mapping, etc.) and biological applications (protein generation, protein structure, and function prediction, etc.), to name a few.

An important limitation of Transformer networks is that the self-attention block, responsible for signal propagation among sequence elements, has a complexity $O(L^2)$ where L is the input sequence length. Because of this, it's hard to apply Transformers in tasks involving long sequence inputs. Long sequences emerge in various applications, such as long text processing, summarization or generation of coherent long texts, high-resolution image generation or recognition, and generation of protein complexes. We consider the problem of long-sequence Transformer training as a motivation for this thesis.

1.2 Outline and contributions of thesis

In this thesis, we give a comprehensive description of the recent innovation of applying random feature mechanisms for the self-attention approximation in Transformers. This technique significantly improves the efficiency of self-attention computation reducing time

and memory complexity from $O(L^2)$ to $O(LM)$ where $M \ll L$ is the number of random features chosen by the user. This improvement is especially important for applications where the sequence length L is big and computing and storing the $(L \times L)$-sized self-attention matrix is too expensive. To make the approximation as precise and stable as possible, we develop novel random feature mechanisms for the Gaussian kernel beyond the existing trigonometric random features (Section 2.1.3 in Chapter 2). We also describe and analyze a fruitful property of Transformers with random features: the ability to forward- and back-propagate through the whole architecture (not just the self-attention block) with up to a constant memory complexity.

The structure of the thesis is as follows:

- Chapter 2 provides a necessary background for the thesis: random features (including existing trigonometric and orthogonal variants) in machine learning and Transformers.
- Chapter 3 introduces the random feature approximation of the self-attention matrix for both causal and noncausal self-attention. This leads to Performers: efficient Transformer architectures. We discuss algorithmic details and theoretical concentration properties of such approximation. We present an exhaustive experimental evaluation of the method.
- Chapter 4 is where positive random features are introduced: a technique to improve stability of random feature approximation in Transformers. This, together with orthogonality, results in the FAVOR+ mechanism (fast attention via positive orthogonal random features). Extensive theoretical and empirical evaluation is present.
- Chapter 5 is dedicated to sub-linear memory Performer: an algorithm for the forward and backward pass through Performer in $O(1)$ memory complexity. Experiments with Transformer networks are included.
- Chapter 6 discusses chef's random tables: extensions of both trigonometric and positive random feature methods. By optimizing variance over this very broad family of random features, we obtain even sharper self-attention approximations. Again, theoretical and empirical analysis is provided.
- Chapter 7 is reserved for conclusions.

Proofs and additional experimental details can be found in the chapters' appendices.

1.3 Publications

This thesis is based on the following publications (* denotes equal contribution):

1. *Rethinking Attention with Performers.* Krzysztof Choromanski*, Valerii Likhosherstov*, David Dohan*, Xingyou Song*, Andreea Gane*, Tamas Sarlos*, Peter Hawkins*, Jared Davis*, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, Adrian Weller. International Conference on Learning Representations (2021).

2. *Sub-Linear Memory: How to Make Performers SLiM.* Valerii Likhosherstov, Krzysztof Choromanski, Jared Davis, Xingyou Song, Adrian Weller. Neural Information Processing Systems (2021).

3. *Chefs' Random Tables: Non-Trigonometric Random Features.* Valerii Likhosherstov*, Krzysztof Choromanski*, Avinava Dubey*, Frederick Liu*, Tamas Sarlos, Adrian Weller. Neural Information Processing Systems (2022).

4. *From Block-Toeplitz Matrices to Differential Equations on Graphs: towards a General Theory for Scalable Masked Transformers.* Krzysztof Choromanski*, Han Lin*, Haoxian Chen*, Tianyi Zhang, Arijit Sehanobish, Valerii Likhosherstov, Jack Parker-Holder, Tamas Sarlos, Adrian Weller, Thomas Weingarten. International Conference on Machine Learning (2022).

5. *Hybrid Random Features.* Krzysztof Choromanski*, Haoxian Chen*, Han Lin*, Yuanzhe Ma*, Arijit Sehanobish*, Deepali Jain, Michael S Ryoo, Jake Varley, Andy Zeng, Valerii Likhosherstov, Dmitry Kalashnikov, Vikas Sindhwani, Adrian Weller. International Conference on Learning Representations (2022).

Apart from that, the following papers were published during author's PhD course:

1. *PolyViT: Co-Training Vision Transformers on Images, Videos and Audio.* Valerii Likhosherstov*, Anurag Arnab*, Krzysztof Choromanski, Mario Lucic, Yi Tay, Adrian Weller, Mostafa Dehghani. Transactions on Machine Learning Research (2022).

2. *Debiasing a First-Order Heuristic for Approximate Bi-Level Optimization.* Valerii Likhosherstov*, Xingyou Song*, Krzysztof Choromanski, Jared Davis, Adrian Weller. International Conference on Machine Learning (2021).

3. *CWY Parametrization: a Solution for Parallelized Optimization of Orthogonal and Stiefel Matrices.* Valerii Likhosherstov*, Jared Davis*, Krzysztof Choromanski, Adrian Weller. International Conference on Artificial Intelligence and Statistics (2021).

4. *An Ode to an ODE.* Krzysztof Choromanski*, Jared Quincy Davis*, Valerii Likhosherstov*, Xingyou Song, Jean-Jacques Slotine, Jacob Varley, Honglak Lee, Adrian Weller, Vikas Sindhwani. Neural Information Processing Systems (2020).

5. *Stochastic Flows and Geometric Optimization on the Orthogonal Group.* Krzysztof Choromanski, David Cheikhi*, Jared Davis*, Valerii Likhosherstov*, Achille Nazaret*, Achraf Bahamou*, Xingyou Song*, Mrugank Akarte, Jack Parker-Holder, Jacob Bergquist, Yuan Gao, Aldo Pacchiano, Tamas Sarlos, Adrian Weller, Vikas Sindhwani. International Conference on Machine Learning (2020).

6. *On the Expressive Power of Self-Attention Matrices.* Valerii Likhosherstov*, Krzysztof Choromanski*, Adrian Weller. Thirty-Seventh AAAI Conference on Artificial Intelligence (2023).

Chapter 2

Background

In this chapter, we will discuss two important prerequisites for this thesis: random features for the Gaussian kernel (Section 2.1) and Transformer networks (Section 2.2).

2.1 Random features for the Gaussian kernel

We define the Gaussian kernel and a corresponding matrix, then we define the notion of random features and discuss its special existing instantiation: trigonometric random features. Then, we talk about orthogonal random features, a technique to reduce variance of vanilla random features. Finally, we overview the literature on random feature features and their applications.

2.1.1 Gaussian kernel matrix

By $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, where d is integer, we denote a *Gaussian kernel*. Namely, for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{y}\|^2\right).$$

Suppose L, L' are integers (L' will coincide with L almost always in this thesis) and two sets $\{\mathbf{x}^{(i)} \in \mathbb{R}^d\}_{i=1}^{L}$, $\{\mathbf{y}^{(j)} \in \mathbb{R}^d\}_{j=1}^{L'}$ are given. Then, *Gaussian kernel matrix* for these two sets is defined as $\mathscr{K} = (K(\mathbf{x}^{(i)}, \mathbf{y}^{(j)}))_{i,j=1}^{L,L'} \in \mathbb{R}^{L \times L'}$. We will be interested in right-multiplying $\mathscr{K}$ by a given matrix $\mathbf{C} \in \mathbb{R}^{L' \times n}$ ($\mathscr{K} \times \mathbf{C}$) as efficiently as possible. The naive algorithm constructs the matrix $\mathscr{K}$ first in $O(LL'd)$ time and then computes the matrix product $\mathscr{K} \times \mathbf{C}$ in $O(LL'n)$ time resulting in $O(LL'(d+n))$ total complexity. When L and L' are big, this complexity is intractable. Therefore, we will be looking for a subquadratic algorithm i.e. without storing the matrix $\mathscr{K}$ in memory and without the the LL' factor in the computational

complexity estimate. We will introduce approximations for solving this problem. With an intention to emphasize that $\mathscr{K}$ is not stored in memory in our efficient methods, we often call it a *linear map* applied to some $\mathbf{C}$.

Efficient estimation of Gaussian kernel linear mappings is important in the following "classical" machine learning methods:

1. **Kernel regression** (Nadaraya, 1964; Watson, 1964). Training data consists of L objects $(\mathbf{o}^{(1)}, \mathbf{r}^{(1)}), \ldots, (\mathbf{o}^{(L)}, \mathbf{r}^{(L)})$ where, for all $1 \leq i \leq L$, $\mathbf{o}^{(i)} \in \mathbb{R}^d$, $\mathbf{r}^{(i)} \in \mathbb{R}^n$. Test data consists of L' observed vectors $\mathbf{o}^{*(1)}, \ldots, \mathbf{o}^{*(L')} \in \mathbb{R}^d$ and the goal is to predict corresponding $\mathbf{r}^{*(1)}, \ldots, \mathbf{r}^{*(L')} \in \mathbb{R}^n$. This is done as follows: for each $1 \leq j \leq L'$,

$$\mathbf{r}_*^{(j)} = \frac{\sum_{i=1}^{L} K(\gamma \mathbf{o}^{*(j)}, \gamma \mathbf{o}^{(i)}) \mathbf{r}^{(i)}}{\sum_{i=1}^{L} K(\gamma \mathbf{o}^{*(j)}, \gamma \mathbf{o}^{(i)})} = \frac{(\mathscr{K}\mathbf{C})_{j,:}}{\mathscr{K}\mathbf{1}_{L'}} \tag{2.1}$$

   where $\gamma > 0$ is a hyperparameter, $\mathscr{K}$ is a Gaussian kernel matrix defined for $\{\mathbf{x}^{(i)} = \gamma \mathbf{o}^{(i)}\}$, $\{\mathbf{y}^{(j)} = \gamma \mathbf{o}^{*(j)}\}$ and $\mathbf{C}$ is an $L' \times n$ matrix with $\mathbf{C}_{j,:} = \mathbf{r}^{*(j)}$ for all $1 \leq j \leq L'$. Computation of (2.1) reduces to multiplication $\mathscr{K} \begin{bmatrix} \mathbf{C} & \mathbf{1}_{L'} \end{bmatrix}$ followed by division.

2. **Kernel support vector machines (Kernel SVM)** (Cristianini and Shawe-Taylor, 2000). In the case of binary classification, the training data consists of L objects $(\mathbf{o}^{(1)}, r^{(1)}), \ldots, (\mathbf{o}^{(L)}, r^{(L)})$ where $\mathbf{o}^{(i)} \in \mathbb{R}^d$ and $r^{(i)} \in \{-1, 1\}$. Kernel SVM equipped with the Gaussian kernel numerically solves the following dual problem:

$$\begin{gathered} \min_{\boldsymbol{\alpha} \in \mathbb{R}^L} \frac{1}{2} \boldsymbol{\alpha}^\top \operatorname{diag}(\mathbf{r}) \mathscr{K} \operatorname{diag}(\mathbf{r}) \boldsymbol{\alpha} - \mathbf{1}_L^\top \boldsymbol{\alpha} \\ \text{s. t. } \mathbf{r}^\top \boldsymbol{\alpha} = 0, \\ \forall 1 \leq i \leq L : 0 \leq \boldsymbol{\alpha}_i \leq C^{\mathrm{SVM}} \end{gathered} \tag{2.2}$$

   where $C^{\mathrm{SVM}} > 0$ is a hyperparameter, $\mathbf{r} = (r^{(i)})_{i=1}^L \in \mathbb{R}^L$ and $\mathscr{K}$ is a Gaussian kernel matrix computed for $\mathbf{x}^{(i)} = \mathbf{y}^{(i)} = \mathbf{o}^{(i)}$, $1 \leq i, j \leq L$, $L = L'$. Computing the objective (2.2) reduces to the evaluation of $\mathscr{K}\mathbf{c}$ where $\mathbf{c} = \operatorname{diag}(\mathbf{r})\boldsymbol{\alpha}$.

In this thesis, we find a novel application for the efficient computation of $\mathscr{K}\mathbf{C}$ which lies in the area of Transformer networks (see Chapter 3). Meanwhile, we start with the description of a prominent random feature approach for the efficient approximation of $\mathscr{K}\mathbf{C}$.

2.1.2 Random features for the Gaussian kernel

Random features is a technique for decomposing the kernel $K(\mathbf{x},\mathbf{y})$ into a product of two terms where each depends either on $\mathbf{x}$ or on $\mathbf{y}$. Since it cannot be done exactly, we rely on a randomized approximation. More technically, the definition of random features is as follows:

Definition 1. *By random features (RFs) for the Gaussian kernel, we call a triple $(p, f^{(1)}, f^{(2)})$ where p is a probability distribution over $\mathbb{R}^d$ and $f^{(i)} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$, $i \in \{1,2\}$, are such that, for all $\mathbf{x} \in \mathbb{R}^d, \mathbf{y} \in \mathbb{R}^d$,*

$$K(\mathbf{x},\mathbf{y}) = \mathbb{E}_{p(\boldsymbol{\omega})} \operatorname{Re}\left(f^{(1)}(\boldsymbol{\omega},\mathbf{x}) f^{(2)}(\boldsymbol{\omega},\mathbf{y})\right). \tag{2.3}$$

Suppose $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(M)}$ are independently distributed according to $p(\boldsymbol{\omega})$. Then for each $\boldsymbol{\omega} = \boldsymbol{\omega}^{(m)}$ we can apply (2.3):

$$\begin{aligned} K(\mathbf{x},\mathbf{y}) &= \mathbb{E}_{p(\boldsymbol{\omega}^{(1)})} \operatorname{Re}\left(f^{(1)}(\boldsymbol{\omega},\mathbf{x}) f^{(2)}(\boldsymbol{\omega}^{(1)},\mathbf{y})\right) \\ &= \cdots = \mathbb{E}_{p(\boldsymbol{\omega}^{(M)})} \operatorname{Re}\left(f^{(1)}(\boldsymbol{\omega},\mathbf{x}) f^{(2)}(\boldsymbol{\omega}^{(M)},\mathbf{y})\right). \end{aligned}$$

Hence, for the mean of M terms in the equation above we also have an identity with $K(\mathbf{x},\mathbf{y})$:

$$\begin{aligned} K(\mathbf{x},\mathbf{y}) &= \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{p(\boldsymbol{\omega}^{(m)})} \operatorname{Re}\left(f^{(1)}(\boldsymbol{\omega}^{(m)},\mathbf{x}) f^{(2)}(\boldsymbol{\omega}^{(m)},\mathbf{y})\right) \\ &= \mathbb{E} \operatorname{Re}\left(\sum_{m=1}^{M} M^{-1/2} f^{(1)}(\boldsymbol{\omega}^{(m)},\mathbf{x}) M^{-1/2} f^{(2)}(\boldsymbol{\omega}^{(m)},\mathbf{y})\right) \end{aligned} \tag{2.4}$$

where in the second transition we use linearity of the expectation $\mathbb{E}$ and the real operator $\operatorname{Re}(\cdot)$ to drag them out of the sum. (2.4) can be seen as a "linearization" of $K(\mathbf{x},\mathbf{y})$ as $\operatorname{Re}\left(\hat{\mathbf{x}}^\top \hat{\mathbf{y}}\right)$ where $\hat{\mathbf{x}}, \hat{\mathbf{y}} \in \mathbb{C}^M$ are defined as

$$\hat{\mathbf{x}} = M^{-1/2}(f^{(1)}(\boldsymbol{\omega}^{(m)},\mathbf{x}^{(i)}))_{m=1}^{M}, \quad \hat{\mathbf{y}} = M^{-1/2}(f^{(2)}(\boldsymbol{\omega}^{(m)},\mathbf{y}^{(i)}))_{m=1}^{M}. \tag{2.5}$$

Indeed, $\operatorname{Re}\left(\hat{\mathbf{x}}^\top \hat{\mathbf{y}}\right)$ is an unbiased approximation of $K(\mathbf{x},\mathbf{y})$ ($\mathbb{E}\operatorname{Re}\left(\hat{\mathbf{x}}^\top \hat{\mathbf{y}}\right) = K(\mathbf{x},\mathbf{y})$) which follows directly from (2.4).

The parameter M controls a tradeoff between the variance of the estimator

$$\begin{aligned} \operatorname{Var}\operatorname{Re}\left(\hat{\mathbf{x}}^\top \hat{\mathbf{y}}\right) &= \frac{1}{M^2}\sum_{m=1}^{M} \operatorname{Var}_{p(\boldsymbol{\omega}^{(m)})} \operatorname{Re}\left(f^{(1)}(\boldsymbol{\omega}^{(m)},\mathbf{x}) f^{(2)}(\boldsymbol{\omega}^{(m)},\mathbf{y})\right) \\ &= \frac{1}{M}\operatorname{Var}_{p(\boldsymbol{\omega})} \operatorname{Re}\left(f^{(1)}(\boldsymbol{\omega},\mathbf{x}) f^{(2)}(\boldsymbol{\omega},\mathbf{y})\right) \end{aligned} \tag{2.6}$$

(decays with M) and the amount of computations (increases with M).

Suppose $\{\mathbf{x}^{(i)}\}_{i=1}^{L}$, $\{\mathbf{y}^{(j)}\}_{j=1}^{L'}$, $\mathbf{C} \in \mathbb{R}^{L' \times n}$ are given, and the goal is to approximate $\mathscr{K}\mathbf{C}$. If $M \ll L, L'$, then the linearization $K(\mathbf{x}, \mathbf{y}) \approx \mathrm{Re}\left(\hat{\mathbf{x}}^\top \hat{\mathbf{y}}\right)$ leads to an unbiased low-rank approximation of $\mathscr{K}$, namely

$$\mathscr{K} = \mathbb{E}\mathrm{Re}\left(\mathbf{S}^{(1)} \times (\mathbf{S}^{(2)})^\top\right)$$

where $\mathbf{S}^{(1)} \in \mathbb{C}^{L \times M}$, $\mathbf{S}^{(2)} \in \mathbb{C}^{L' \times M}$,

$$\forall 1 \le i \le L, 1 \le m \le M : \mathbf{S}^{(1)}_{i,m} = M^{-1/2} f^{(1)}(\boldsymbol{\omega}^{(m)}, \mathbf{x}^{(i)}), \tag{2.7}$$

$$\forall 1 \le j \le L', 1 \le m \le M : \mathbf{S}^{(2)}_{j,m} = M^{-1/2} f^{(2)}(\boldsymbol{\omega}^{(m)}, \mathbf{y}^{(j)}). \tag{2.8}$$

Now, for $\mathscr{K}$ as the linear map, we have:

$$\begin{aligned}\mathscr{K}\mathbf{C} &= \left(\mathbb{E}\mathrm{Re}\left(\mathbf{S}^{(1)} \times (\mathbf{S}^{(2)})^\top\right)\right)\mathbf{C} = \mathbb{E}\mathrm{Re}\left(\mathbf{S}^{(1)} \times (\mathbf{S}^{(2)})^\top \times \mathbf{C}\right) \\ &= \mathbb{E}\mathrm{Re}\left(\mathbf{S}^{(1)} \times ((\mathbf{S}^{(2)})^\top \times \mathbf{C})\right)\end{aligned}$$

where we first use linearity of $\mathbb{E}(\mathrm{Re}(\ldots))$ to put $\mathbf{C}$ under the expectation and the real part, and then we use associativity of the matrix product to change the order of multiplications. Algorithm 1 summarizes the sequence of computations for the approximation.

The computational complexity of approximation $\mathscr{K}\mathbf{C} \approx \mathrm{Re}\left(\mathbf{S}^{(1)} \times ((\mathbf{S}^{(2)})^\top \times \mathbf{C})\right)$ is $O((L+L')Md)$ for computing $\mathbf{S}^{(1)}, \mathbf{S}^{(2)}$ (assuming that finding $f^{(\cdot)}(\cdot,\cdot)$ takes $O(d)$ computations which is usually the case) plus $O((L+L')Mn)$ for evaluating two matrix products, resulting in $O((L+L')M(d+n))$. This is a subquadratic approximation since $(L+L')M \ll LL'$ when $M \ll L, L'$. Next, we discuss trigonometric random features, an instantiation of the generic Definition 1.

2.1.3 Trigonometric random features

Historically random features were introduced back in 2007 in (Rahimi and Recht, 2007) where the first type of random features for the Gaussian kernel was proposed. Authors rely on the Bochner's theorem which states:

Theorem 1 (Bochner's theorem (Rudin, 2017))**.** *A continuous shift-invariant kernel $K'(\mathbf{x}, \mathbf{y}) = K'(\mathbf{x} - \mathbf{y})$ on $\mathbb{R}^d$ is positive definite if and only if $K'(\boldsymbol{\delta})$ is the Fourier transform of a non-negative measure.*

Algorithm 1 Algorithm for an unbiased approximation of the Gaussian kernel linear map.

1: **Input:** $\{\mathbf{x}^{(i)} \in \mathbb{R}^d\}_{i=1}^{L}$, $\{\mathbf{y}^{(j)} \in \mathbb{R}^d\}_{i=1}^{L'}$, $\mathbf{C} \in \mathbb{R}^{L \times n}$.
2: **Output:** unbiased approximation of $\mathscr{K}\mathbf{C}$ where $\mathscr{K}$ is the Gaussian kernel matrix computed for $\{\mathbf{x}^{(i)} \in \mathbb{R}^d\}_{i=1}^{L}$ and $\{\mathbf{y}^{(j)} \in \mathbb{R}^d\}_{i=1}^{L'}$.
3: Draw i.i.d. $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(M)} \sim p(\boldsymbol{\omega})$;
4: Set $\mathbf{S}^{(1)} = (f^{(1)}(\boldsymbol{\omega}^{(m)}, \mathbf{x}^{(i)}))_{i,m=1}^{L,M}$; ▷ $O(LMd)$
5: Set $\mathbf{S}^{(2)} = (f^{(2)}(\boldsymbol{\omega}^{(m)}, \mathbf{y}^{(j)}))_{j,m=1}^{L',M}$; ▷ $O(L'Md)$
6: Compute $\mathbf{S}^{(3)} = (\mathbf{S}^{(2)})^\top \times \mathbf{C}$; ▷ $O(L'Mn)$
7: Compute $\mathbf{S}^{(4)} = \mathbf{S}^{(1)} \times \mathbf{S}^{(3)}$; ▷ $O(LMn)$
8: **Return** $\mathrm{Re}\left(\mathbf{S}^{(4)}\right)$.

If the kernel $K'(\cdot,\cdot)$ is properly scaled, this measure is a valid probability distribution. Further, Rahimi and Recht (2007) observe that if $K'(\mathbf{x},\mathbf{y})$ is the Gaussian kernel $K(\mathbf{x},\mathbf{y}) = \exp(-\frac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2)$, this measure is $\exp\left(-\frac{1}{2}\|\boldsymbol{\omega}\|^2\right)$. Namely, for all $\mathbf{x},\mathbf{y} \in \mathbb{R}^d$,

$$\begin{aligned} K(\mathbf{x},\mathbf{y}) = K(\mathbf{y},\mathbf{x}) &= (2\pi)^{-d/2} \int_{\mathbb{R}^d} \exp\left(-\frac{1}{2}\|\boldsymbol{\omega}\|^2\right) \exp(-\mathrm{i}\boldsymbol{\omega}^\top(\mathbf{y}-\mathbf{x}))\mathrm{d}\boldsymbol{\omega} \\ &= (2\pi)^{-d/2} \int_{\mathbb{R}^d} \exp\left(-\frac{1}{2}\|\boldsymbol{\omega}\|^2\right) \exp(\mathrm{i}\boldsymbol{\omega}^\top(\mathbf{x}-\mathbf{y}))\mathrm{d}\boldsymbol{\omega} \end{aligned}$$

where we use symmetry of $K(\cdot,\cdot)$ and the definition of the Fourier transform. Next, we recall the density of the standard multivariate Gaussian distribution $p_{\mathrm{SG}}(\boldsymbol{\omega}) = (2\pi)^{-d/2} \exp\left(-\frac{1}{2}\|\boldsymbol{\omega}\|^2\right)$ and deduce that

$$\begin{aligned} (2\pi)^{-d/2} \int_{\mathbb{R}^d} \exp\left(-\frac{1}{2}\|\boldsymbol{\omega}\|^2\right) \exp(\mathrm{i}\boldsymbol{\omega}^\top(\mathbf{x}-\mathbf{y}))\mathrm{d}\boldsymbol{\omega} = \\ \int_{\mathbb{R}^d} p_{\mathrm{SG}}(\boldsymbol{\omega}) \exp(\mathrm{i}\boldsymbol{\omega}^\top\mathbf{x}) \exp(-\mathrm{i}\boldsymbol{\omega}^\top\mathbf{y})\mathrm{d}\boldsymbol{\omega}. \end{aligned}$$

The right hand side can be expressed as an expectation with respect to $\boldsymbol{\omega} \sim \mathscr{N}(0,1)^d$:

$$\int_{\mathbb{R}^d} p_{\mathrm{SG}}(\boldsymbol{\omega}) \exp(\mathrm{i}\boldsymbol{\omega}^\top\mathbf{x}) \exp(-\mathrm{i}\boldsymbol{\omega}^\top\mathbf{y}) d\boldsymbol{\omega} = \mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}\left(f_{\mathrm{trig}}^{(1)}(\boldsymbol{\omega},\mathbf{x}) f_{\mathrm{trig}}^{(2)}(\boldsymbol{\omega},\mathbf{y})\right)$$

which gives

$$K(\mathbf{x},\mathbf{y}) = \mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}\left(f_{\mathrm{trig}}^{(1)}(\boldsymbol{\omega},\mathbf{x}) f_{\mathrm{trig}}^{(2)}(\boldsymbol{\omega},\mathbf{y})\right).$$

where $f^{(1)}_{\text{trig}}(\omega, \mathbf{x}) = \exp(\mathrm{i}\boldsymbol{\omega}^\top \mathbf{x})$, $f^{(2)}_{\text{trig}}(\boldsymbol{\omega}, \mathbf{x}) = \exp(-\mathrm{i}\boldsymbol{\omega}^\top \mathbf{y})$. Finally, we observe that the left-hand side is real-valued, hence

$$\begin{aligned} K(\mathbf{x},\mathbf{y}) &= \operatorname{Re}\left(K(\mathbf{x},\mathbf{y})\right) = \operatorname{Re}\left(\mathbb{E}_{p_{\text{SG}}(\boldsymbol{\omega})}\left(f^{(1)}_{\text{trig}}(\boldsymbol{\omega},\mathbf{x}) f^{(2)}_{\text{trig}}(\boldsymbol{\omega},\mathbf{y})\right)\right) \\ &= \mathbb{E}_{p_{\text{SG}}(\boldsymbol{\omega})} \operatorname{Re}\left(f^{(1)}_{\text{trig}}(\boldsymbol{\omega},\mathbf{x}) f^{(2)}_{\text{trig}}(\boldsymbol{\omega},\mathbf{y})\right). \end{aligned} \tag{2.9}$$

The last transition follows from the fact that $\operatorname{Re}(\cdot)$ is a linear mapping and hence can be moved inside the expectation.

Representation (2.9) proves that $(p_{\text{SG}}, f^{(1)}_{\text{trig}}, f^{(2)}_{\text{trig}})$ are valid random features for the Gaussian kernel. We refer to them as *trigonometric random features* (TrigRFs) since $\exp(\mathrm{i}\cdot) = \cos(\cdot) + \mathrm{i}\sin(\cdot)$ reduces to computing sines and cosines in practice.

An important property of TrigRFs is that they provide a bounded unbiased approximation. Indeed, $|f^{(1)}_{\text{trig}}(\boldsymbol{\omega},\mathbf{x})|, |f^{(2)}_{\text{trig}}(\boldsymbol{\omega},\mathbf{y})| = 1$. Because of this property, first of all, the variance of TrigRFs is also always bounded (Yu et al., 2016):

$$\operatorname{Var}_{p_{\text{SG}}(\boldsymbol{\omega})} \operatorname{Re}\left(f^{(1)}_{\text{trig}}(\boldsymbol{\omega},\mathbf{x}) f^{(2)}_{\text{trig}}(\boldsymbol{\omega},\mathbf{y})\right) = \frac{1}{2}\left(1 - K(\mathbf{x},\mathbf{y})^2\right)^2 \leq \frac{1}{2} \tag{2.10}$$

since $K(\mathbf{x},\mathbf{y}) \leq 1$. Further, the following tight concentration result can be proven:

Theorem 2 (Claim 1 from Rahimi and Recht (2007))**.** *Let $R, \varepsilon > 0$ and $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(M)}$ be i.i.d. samples from $p_{\text{SG}}(\boldsymbol{\omega})$. Then:*

$$\mathbb{P}\left(\sup_{\|\mathbf{x}\|,\|\mathbf{y}\| \leq R} |K(\mathbf{x},\mathbf{y}) - \operatorname{Re}\left(\hat{\mathbf{x}}^\top \hat{\mathbf{y}}\right)| \geq \varepsilon\right) \leq \mathscr{C}\left(\frac{dR}{\varepsilon}\right)^2 \exp\left(-\frac{M\varepsilon^2}{4(d+2)}\right)$$

where $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ are defined according to (2.5) and $\mathscr{C}$ is an absolute constant. Hence, with any constant probability, $\sup_{\|\mathbf{x}\|,\|\mathbf{y}\| \leq R} |K(\mathbf{x},\mathbf{y}) - \operatorname{Re}\left(\hat{\mathbf{x}}^\top \hat{\mathbf{y}}\right)| < \varepsilon$ when $M = O\left(\frac{d}{\varepsilon^2} \log \frac{dR}{\varepsilon}\right)$.

According to Theorem 2, when the magnitude of input vectors is bounded by D and the error of approximation is bounded by ε, one needs only $M = O(d \log d)$ random features to get approximation of $K(\cdot,\cdot)$ for all possible inputs with a given constant probability. In the next section, we discuss an approach which further improves TrigRFs and achieves an even smaller error of approximation.

2.1.4 Orthogonal random features

Yu et al. (2016) propose to sample $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(M)}$ not as independent vectors, but as *orthogonal* vectors which are still marginally drawn from $\mathcal{N}(0,1)^d$. Denote

$$\boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{\omega}^{(1)} & \ldots & \boldsymbol{\omega}^{(M)} \end{bmatrix} \in \mathbb{R}^{d \times M}.$$

For i.i.d. vectors, $\boldsymbol{\Omega}$ is defined as a random matrix sampled from $\mathcal{N}(0,1)^{d \times M}$. In the orthogonal case, it is defined as a concatenation of $\approx M/d$ $(d \times d)$-sized rescaled orthogonal matrices. Algorithm 2 gives a formal description for computing $\boldsymbol{\Omega}$. Denote

$$\hat{\mathbf{x}}_{\text{ort}} = M^{-1/2}(f^{(1)}(\boldsymbol{\Omega}_{:,m}, \mathbf{x}^{(i)}))_{m=1}^{M}, \quad \hat{\mathbf{y}}_{\text{ort}} = M^{-1/2}(f^{(2)}(\boldsymbol{\Omega}_{:,m}, \mathbf{y}^{(i)}))_{m=1}^{M}. \tag{2.11}$$

where $\boldsymbol{\Omega}$ is sampled according to the Algorithm 2. Observe that each column of $\boldsymbol{\Omega}$ is marginally distributed as $\mathcal{N}(0,1)^d$. For instance, $\boldsymbol{\Omega}_{:,1}$ is a product of $\boldsymbol{\Omega}^{(1)}_{:,1}$, which is marginally distributed uniformly on a sphere $\mathbb{S}^{d-1}$, and $\boldsymbol{\chi}_1$, which is independent from $\boldsymbol{\Omega}^{(1)}_{:,1}$ and is distributed as a magnitude of a vector from $\mathcal{N}(0,1)^d$. Same for all other columns of $\boldsymbol{\Omega}$. Hence, approximation $K(\mathbf{x},\mathbf{y}) \approx \operatorname{Re}\left(\hat{\mathbf{x}}_{\text{ort}}^{\top}\hat{\mathbf{y}}_{\text{ort}}\right)$ is still unbiased:

$$\mathbb{E}_{\boldsymbol{\Omega}} \operatorname{Re}\left(\hat{\mathbf{x}}_{\text{ort}}^{\top}\hat{\mathbf{y}}_{\text{ort}}\right) = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{\Omega}_{:,m}} \operatorname{Re}\left(f^{(1)}(\boldsymbol{\Omega}_{:,m}, \mathbf{x}) f^{(2)}(\boldsymbol{\Omega}_{:,m}, \mathbf{y})\right) = \frac{1}{M}\sum_{m=1}^{M} K(\mathbf{x},\mathbf{y}) = K(\mathbf{x},\mathbf{y}).$$

We refer to random features using Ω sampled according to Algorithm 2 as *orthogonal random features* (ORFs).

As for the variance, in the i.i.d. features case it is (according to (2.6, 2.10))

$$\operatorname{Var}\operatorname{Re}\left(\hat{\mathbf{x}}^{\top}\hat{\mathbf{y}}\right) = \frac{1}{2M}(1 - K(\mathbf{x},\mathbf{y})^2)^2. \tag{2.12}$$

The following theorem quantifies the improvement of orthogonal RFs

Theorem 3 (Yu et al. (2016))**.** *There exists a function $F_{\text{trig}} : \mathbb{R} \to \mathbb{R}$ such that, if $M \leq d$, then*

$$\operatorname{Var}\operatorname{Re}\left(\hat{\mathbf{x}}_{\text{ort}}^{\top}\hat{\mathbf{y}}_{\text{ort}}\right) \leq \operatorname{Var}\operatorname{Re}\left(\hat{\mathbf{x}}^{\top}\hat{\mathbf{y}}\right) - \frac{M-1}{2Md}K(\mathbf{x},\mathbf{y})^2\|\mathbf{x}-\mathbf{y}\|^4 + \frac{F_{\text{trig}}(\|\mathbf{x}-\mathbf{y}\|)}{d^2}$$

If $M > d$ and M/d is integer, then

$$\operatorname{Var}\operatorname{Re}\left(\hat{\mathbf{x}}_{\text{ort}}^{\top}\hat{\mathbf{y}}_{\text{ort}}\right) \leq \operatorname{Var}\operatorname{Re}\left(\hat{\mathbf{x}}^{\top}\hat{\mathbf{y}}\right) - \frac{d-1}{2Md}K(\mathbf{x},\mathbf{y})^2\|\mathbf{x}-\mathbf{y}\|^4 + \frac{F_{\text{trig}}(\|\mathbf{x}-\mathbf{y}\|)}{Md}.$$

When d or M is big enough, the term $F_{\mathrm{trig}}(\|\mathbf{x}-\mathbf{y}\|)/d^2$ or $F_{\mathrm{trig}}(\|\mathbf{x}-\mathbf{y}\|)/Md$ vanishes and $\mathrm{Var}\,\mathrm{Re}\left(\hat{\mathbf{x}}_{\mathrm{ort}}^{\top}\hat{\mathbf{y}}_{\mathrm{ort}}\right)$ becomes strictly smaller than $\mathrm{Var}\,\mathrm{Re}\left(\hat{\mathbf{x}}^{\top}\hat{\mathbf{y}}\right)$ – the variance of the i.i.d. feature case. This improvement comes at the cost of an additional $O(Md^2)$ term in the complexity estimate for running M/d QR decompositions of size $d \times d$. Though, in the considered scenario $L, L' \gg d$ and, therefore, $(L+L')M(d+n) \gg Md^2$, i.e. this addition is negligible.

Algorithm 2 Algorithm for sampling Ω for orthogonal RFs. Total complexity: $O((\tau+1)d^3) = O((M/d)d^3) = O(Md^2)$. $\mathbb{N}$ denotes a set of *positive* integers.

1: **Input:** $d, M \in \mathbb{N}$.
2: **Output:** $\mathbf{\Omega} = \left[\boldsymbol{\omega}^{(1)} \quad \ldots \quad \boldsymbol{\omega}^{(M)}\right] \in \mathbb{R}^{d \times M}$.
3: Let $\tau = \lceil M/d \rceil$, $\nu = M - d\tau$;
4: Draw τ i.i.d. random orthogonal matrices $\mathbf{\Omega}^{(1)}, \ldots, \mathbf{\Omega}^{(\tau)} \in \mathbb{R}^{d \times d}$, where $\mathbf{\Omega}^{(i)}$ is drawn by taking "Q" part of the QR factorization of the matrix sampled from $\mathcal{N}(0,1)^{d \times d}$;
5: Draw $\boldsymbol{\chi} \in \mathbb{R}^M$ – a vector with i.i.d. entries sampled from $\chi(d)$ distribution;
6: **Return** $\left[\mathbf{\Omega}^{(1)} \quad \ldots \quad \mathbf{\Omega}^{(\tau-1)} \quad \mathbf{\Omega}^{(\tau)}_{:,:\nu}\right] \times \mathrm{diag}(\boldsymbol{\chi})$.

2.1.5 Literature overview: random features in machine learning

The idea that nonlinear mappings of the random-weight linear combinations of data features can be used to linearize various nonlinear similarity functions transformed kernel methods. This led to the development of random feature techniques; and the new field of scalable kernel algorithms, introduced in the paper trilogy (Rahimi and Recht, 2007, 2008a,b), was born. Random features were subsequently used in many applications, ranging from kernel and function-to-function regression (Avron et al., 2017; Laparra et al., 2015; Oliva et al., 2015), SVM algorithms (Sun et al., 2018) to operator-valued and semigroup kernels (Minh, 2016; Yang et al., 2014), neural networks (Cho and Saul, 2009; Gonon, 2021; Han et al., 2021; Xie et al., 2019) and even differentially-private ML algorithms (Chaudhuri et al., 2011), as well as nonparametric adaptive control (Boffi et al., 2021). Random features are a subject of much theoretical analysis (Li et al., 2021; Sriperumbudur and Szabó, 2015; Sutherland and Schneider, 2015; Yang et al., 2012). To approximate shift invariant (e.g. Gaussian, Cauchy or Laplace) and softmax kernels, random features rely on the trigonometric nonlinear mappings provided directly by Bochner's theorem (Minh, 2016).

Orthogonal matrices as random projections result in tighter approximations than unconstrained projections (Choromanski et al., 2017a; Lin et al., 2020). Ensembles of orthogonal random projections were shown to provide much better concentration results for the estimators relying on them in various other contexts, in particular: kernel approximation (Bojarski

et al., 2017; Choromanska et al., 2016; Choromanski and Sindhwani, 2016; Choromanski et al., 2017a), estimation of the gradients of Gaussian smoothings with evolution strategy methods (Choromanski et al., 2018b), kernel ridge regression techniques (Choromanski et al., 2018a) and sliced Wasserstein distance estimation (Rowland et al., 2019).

Our next background topic is Transformers: prominent deep learning architectures where we will apply improved random feature techniques in the subsequent chapters.

2.2 Transformers and self-attention

Proposed in 2017 for neural machine translation (Vaswani et al., 2017), Transformers quickly became widespread in the field of deep learning as state of the art neural network architectures for many problems. Transformer backbone is designed for sequence processing and can be used to solve sequence-to-sequence, language modelling and classification problems in different application domains, including natural language processing (Brown et al., 2020; Devlin et al., 2019; Radford et al., 2019; Vaswani et al., 2017), computer vision (Arnab et al., 2021; Dosovitskiy et al., 2021; Sun et al., 2019), protein processing in biology (Elnaggar et al., 2019; Madani et al., 2020; Rives et al., 2019) and beyond.

In this part of the chapter, we define the Transformer architecture and self-attention, an important part of Transformers responsible for signal propagation along the sequence. Then, we discuss a number of Transformer applications: language modelling, classification and vision Transformers. Finally, we overview literature on Transformer applications.

2.2.1 Transformer architecture

Transformers process sequential data fed as the input to the architecture. Let L denote the size of the input sequence. First, each element of the sequence, which can be either a discrete or a continuous object, is processed into the vector of length d_{hid} referred to as *token*. Let $\mathbf{X}^{(0)} \in \mathbb{R}^{L \times d_{\text{hid}}}$ denote a matrix where each row $\mathbf{X}_i^{(0)} \in \mathbb{R}^{d_{\text{hid}}}$ corresponds to i'th *token*. Then *Transformer* is defined as a parametrized mapping $\mathbf{X}^{(0)} \rightarrow \mathbf{X}^{(1)} \rightarrow \cdots \rightarrow \mathbf{X}^{(s)} \rightarrow \mathbf{X}^{(\text{out})}$ where $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(s)} \in \mathbb{R}^{L \times d_{\text{hid}}}$ are *intermediate representations* and $\mathbf{X}^{(\text{out})} \in \mathbb{R}^{L \times d_{\text{out}}}$ is a Transformer *output*.

Each mapping $X^{(r-1)} \rightarrow X^{(r)}$ is called a Transformer *layer* and it consists of two parts: *multi-head self-attention*

$$\overline{\mathbf{X}}^{(r-1)} = \text{LN}(\text{MultiHead-Att}(\mathbf{X}^{(r-1)}) + \mathbf{X}^{(r-1)} \tag{2.13}$$

and *feed-forward network*

$$\mathbf{X}^{(r)} = \mathrm{LN}(\mathrm{FFN}(\overline{\mathbf{X}}^{(r-1)})) + \overline{\mathbf{X}}^{(r-1)}. \tag{2.14}$$

Here, LN denotes *layer normalization* (Ba et al., 2016) applied separately per each row of its input matrix. Layer normalization shifts and scales the input so that its mean is zero and variance is one, and then it additionally applies a trainable shift and scale to each position of the output. More formally, if $\mathbf{z} \in \mathbb{R}^{d_{\mathrm{hid}}}$ is an input, then $\mathrm{LN} : \mathbb{R}^{d_{\mathrm{hid}}} \to \mathbb{R}^{d_{\mathrm{hid}}}$ is defined as follows:

$$\forall 1 \leq l \leq d_{\mathrm{hid}} : \mathrm{LN}(\mathbf{z})_l = \frac{1}{\sigma^{\mathrm{LN}}} \mathbf{g}^{\mathrm{LN}}_l (\mathbf{z}_l - \mu^{\mathrm{LN}}) + \mathbf{b}^{\mathrm{LN}}_l,$$
$$\mu^{\mathrm{LN}} = \frac{1}{d_{\mathrm{hid}}} \sum_{l=1}^{d_{\mathrm{hid}}} \mathbf{z}_l, \quad \sigma^{\mathrm{LN}} = \sqrt{\frac{1}{d_{\mathrm{hid}}} \sum_{l=1}^{d_{hid}} (\mathbf{z}_l - \mu^{\mathrm{LN}})^2},$$

where $\mathbf{g}^{\mathrm{LN}}, \mathbf{b}^{\mathrm{LN}} \in \mathbb{R}^{d_{\mathrm{hid}}}$ are trainable parameters. Layer normalization and *skip connections* (second additive terms in (2.13,2.14)) are needed to improve stability during training of the Transformer.

Multi-head self-attention is defined as h *self-attention heads* applied in parallel. The input to each head is a linear transformation of $\mathbf{X}^{(r-1)}$:

$$\mathrm{MultiHead\text{-}Att}(\mathbf{X}^{(r-1)}) = [\mathbf{H}^{(1)} \dots \mathbf{H}^{(h)}]\mathbf{W}_{\mathrm{O}},$$
$$\forall 1 \leq l \leq h : \mathbf{H}^{(l)} = \mathrm{Att}_{\square}(\mathbf{X}^{(r-1)}\mathbf{W}^{(l)}_{\mathrm{Q}}, \mathbf{X}^{(r-1)}\mathbf{W}^{(l)}_{\mathrm{K}}, \mathbf{X}^{(r-1)}\mathbf{W}^{(l)}_{\mathrm{V}}) \tag{2.15}$$

where $\mathrm{Att}_{\square}(\cdot,\cdot,\cdot)$ is a self-attention operation defined in the next section, $\square \in \{\rightarrow, \leftrightarrow\}$ denotes the type of self-attention (also defined in the next section), $\mathbf{W}^{(l)}_{\mathrm{Q}}, \mathbf{W}^{(l)}_{\mathrm{K}}, \mathbf{W}^{(l)}_{\mathrm{V}} \in \mathbb{R}^{d_{\mathrm{hid}} \times d}$ are trainable projection matrices used for obtaining input matrices passed into self-attention, $\mathbf{W}_{\mathrm{O}} \in \mathbb{R}^{d_{\mathrm{hid}} \times d_{\mathrm{hid}}}$ is a trainable output transformation matrix, $d = d_{\mathrm{hid}}/h$ is the self-attention dimension which is also called *query dimension*. Usually, $h = d_{\mathrm{hid}}/64$ and, consequently, the query dimension is $d = 64$. Feed-forward network is defined as a neural network with a single hidden layer applied independently to each element of the sequence:

$$\forall 1 \leq i \leq L : \mathrm{FFN}(\overline{\mathbf{X}}^{(r-1)})_i = \mathbf{W}^{(2)}\mathrm{GeLU}[\mathbf{W}^{(1)}\overline{\mathbf{X}}^{(r-1)}_i + \mathbf{b}^{(1)}] + \mathbf{b}^{(2)} \in \mathbb{R}^{d_{\mathrm{hid}}}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{d_{\mathrm{ff}} \times d_{\mathrm{hid}}}$, $\mathbf{b}^{(1)} \in \mathbb{R}^{d_{\mathrm{ff}}}$, $\mathbf{W}^{(2)} \in \mathbb{R}^{d_{\mathrm{hid}} \times d_{\mathrm{ff}}}$, $\mathbf{b}^{(2)} \in \mathbb{R}^{d_{\mathrm{hid}}}$ are trainable parameters, d_{ff} is the width of the hidden layer, usually $d_{\mathrm{ff}} = 4d_{\mathrm{hid}}$. GeLU denotes Gaussian error linear

unit (Hendrycks and Gimpel, 2016), an activation function applied elementwise:

$$\mathrm{GeLU}(z) = \frac{1}{2}z\left(1+\mathrm{erf}\left(z/\sqrt{2}\right)\right).$$

In the original Transformer (Vaswani et al., 2017), $\mathrm{ReLU}(z) = \max(0,z)$ activation (Fukushima, 1975) was used instead of GeLU, however later GeLU was shown to perform better (Radford et al., 2019) and it is a widely accepted default choice in Transformers now.

Finally, $\mathbf{X}^{(\mathrm{out})}$ is obtained through a linear mapping of $\mathbf{X}^{(s)}$:

$$\forall 1 \le i \le L : \mathbf{X}_i^{(\mathrm{out})} = \mathbf{W}^{(\mathrm{out})}\mathbf{X}_i^{(s)} + \mathbf{b}^{(\mathrm{out})}$$

where $\mathbf{W}^{(\mathrm{out})} \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{hid}}}$ and $\mathbf{b}^{(\mathrm{out})} \in \mathbb{R}^{d_{\mathrm{out}}}$, d_{out} is the dimension of the output.

In all instantiations of MultiHead-Att, LN, FFN, we assume that parameters are different.

2.2.2 Self-attention mechanism

Now we are in the position to define self-attention: an important part of Transformer architecture which is responsible for signal propagation across elements of the sequence. *Self-attention* is a mapping which accepts three matrices $\mathbf{Q},\mathbf{K},\mathbf{V} \in \mathbb{R}^{L\times d}$ as input where d is the *query dimension*. Matrices $\mathbf{Q},\mathbf{K},\mathbf{V}$ can be interpreted as *queries*, *keys* and *values*, and self-attention as a lookup into the continuous dictionary defined by $\mathbf{Q},\mathbf{K},\mathbf{V}$ respectively. As previously discussed, $\mathbf{Q},\mathbf{K}$ and $\mathbf{V}$ are results of linear mappings performed on rows of $\mathbf{X}^{(r-1)}$ which explains the "self-" prefix in self-attention, i.e. the model is attending to itself.

We consider two types of self-attention. *Bidirectional (or noncausal) self-attention* has the following form:

$$\mathrm{Att}_{\leftrightarrow}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{D}^{-1}\mathbf{A}\mathbf{V} \in \mathbb{R}^{L\times d}, \quad \mathbf{A} = \exp[\mathbf{Q}\mathbf{K}^\top/\sqrt{d}] \in \mathbb{R}^{L\times L}, \quad \mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{1}_L). \tag{2.16}$$

where by $\rho[\cdot]$ we denote elementwise application of the unary function $\rho(\cdot)$ ($\rho = \exp$ in the expression above). $\mathbf{D}^{-1}\mathbf{A}$ is a *left-stochastic* matrix, meaning that its rows are nonnegative and sum to 1. Each row of $\mathbf{D}^{-1}\mathbf{A}$ is the result of a *softmax* operation which takes elementwise exponent of the input and normalizes it. The $\sqrt{d}$-scaling term inside the exponent is needed for the stability during training (Vaswani et al., 2017). Hence, the weight of the query-key pair in the differentiable dictionary is defined by the value $\exp(\mathbf{Q}_i^\top \mathbf{K}_j/\sqrt{d})$. Another important type of self-attention is *unidirectional (or causal) self-attention* which has the form:

$$\mathrm{Att}_{\rightarrow}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{A}}\mathbf{V} \in \mathbb{R}^{L\times d}, \quad \widetilde{\mathbf{A}} = \mathrm{tril}(\mathbf{A}), \quad \widetilde{\mathbf{D}} = \mathrm{diag}(\widetilde{\mathbf{A}}\mathbf{1}_L), \tag{2.17}$$

where $\mathbf{A}$ is defined in the same way as in (2.16) and $\text{tril}(\mathbf{A})$ is a copy of $\mathbf{A}$ but with entries above the main diagonal zeroed out.

The most expensive parts of computing noncausal attention (2.16) are evaluating the product $\mathbf{Q} \times \mathbf{K}^\top \in \mathbb{R}^d$ and then the product $\mathbf{A} \times \mathbf{V}$, both taking $O(L^2 d)$ time. Same is true for the causal attention (2.17). Hence, self-attention is intractable when the sequence length is big. This problem and its solution will be discussed in the next chapters. In the remainder of the section, we discuss some prominent applications of Transformer networks which will be used for experimental evaluation in the subsequent chapters.

2.2.3 Language modelling with Transformers

Language modelling is an important problem in machine learning which consists in fitting a probability distribution over a corpus of sequences. Let Σ be a finite set (a *vocabulary*), then each sequence $\mathbf{p}$ in the corpus is an ordered set of *words* from the alphabet of length L, i.e. $\mathbf{p} \in \Sigma^L$.

Autoregressive language modelling defines the probability to generate sequence $\mathbf{p}$ by Transformer as follows:

$$\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{p}) = \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{p}_1) \times \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{p}_2|\mathbf{p}_1) \times \cdots \times \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{p}_L|\mathbf{p}_1, \ldots, \mathbf{p}_{L-1}),$$

where, for each $1 \leq i \leq L$,

$$\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{p}_i|\mathbf{p}_1, \ldots, \mathbf{p}_{i-1}) = \frac{\exp(\mathbf{X}^{(\text{out})}_{i,\mathbf{p}_i})}{\sum_{l=1}^{d_{\text{out}}} \exp(\mathbf{X}^{(\text{out})}_{i,l})} \tag{2.18}$$

is defined by the vector $\mathbf{X}^{(\text{out})}_i$ as logits, where $\mathbf{X}^{(\text{out})} \in \mathbb{R}^{L \times d_{\text{out}}}$, $d_{\text{out}} = |\Sigma|$, is the output of the Transformer with causal self-attention, i.e. $\square =\!\!\rightarrow$ in (2.15). Vector $\boldsymbol{\theta} \in \mathbb{R}^{d_{\text{param}}}$ denotes the set of all parameters of this Transformer and as the input this Transformer receives *tokenized* sequence $\mathbf{p}$, meaning that

$$\forall 1 \leq i \leq L : \mathbf{X}^{(0)}_i = \mathbf{emb}_{\mathbf{p}_i} + \mathbf{posemb}_i \tag{2.19}$$

where $\{\mathbf{emb}_z \in \mathbb{R}^{d_{\text{hid}}}\}_{z \in \Sigma}$ is a set of trainable *word embeddings* and $\{\mathbf{posemb}_i \in \mathbb{R}^{d_{\text{hid}}}\}_{1 \leq i \leq L}$ is a set of trainable *positional embeddings*.

By the definition of causal self-attention, for each $1 \leq i \leq L, 1 \leq r \leq s$, $\mathbf{X}^{(r)}_i$ only depends on $\{\mathbf{X}^{(r-1)}_j\}_{1 \leq j \leq i}$. Hence, $\mathbf{X}^{(\text{out})}_i$ only depends on $\{\mathbf{p}_j\}_{1 \leq j \leq i}$ which makes the definition of $\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{p}_i|\mathbf{p}_1, \ldots, \mathbf{p}_{i-1})$ valid. During training, the loss $\mathscr{L}$, minimized with respect to $\boldsymbol{\theta}$ via

minibatch gradient descent (Bottou et al., 2016), is the scaled negative log-likelihood of observing the sequence $\mathbf{p}$ from the input:

$$\mathscr{L} = -\frac{1}{L}\log \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{p}) = -\frac{1}{L}\sum_{i=1}^{L}\log \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{p}_i|\mathbf{p}_1,\dots,\mathbf{p}_{i-1}). \tag{2.20}$$

Autoregressive language modelling can be used for generative modelling on texts (Brown et al., 2020; Radford et al., 2019), images (Bello et al., 2019) by flattening them into sequences of color channels or proteins represented as sequences of amino acids (Madani et al., 2020; Rives et al., 2019).

Alternatively, *masked* language modelling (Devlin et al., 2019) employs a Transformer with noncausal self-attention ($\square = \leftrightarrow$ in (2.15)). Given an input sequence $\mathbf{p}$, it is first modified into $\widehat{\mathbf{p}}$ by randomly replacing some of the words by UNK. Then, this sequence is passed into the Transformer in the same way as in autoregressive modelling (2.19) (UNK is treated as an additional word in the vocabulary Σ). The optimization goal is to minimize the loss

$$\mathscr{L}^{\mathrm{M}} = -\frac{1}{L_{\mathrm{UNK}}}\sum_{1\le i\le L, \widehat{\mathbf{p}}_i=\mathrm{UNK}}\log \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{p}_i|\widehat{\mathbf{p}}), \quad L_{\mathrm{UNK}} = |\{1\le i\le L\,|\,\widehat{\mathbf{p}}_i = \mathrm{UNK}\}| \tag{2.21}$$

where the distribution $\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{p}_i|\widehat{\mathbf{p}})$ is modelled by $\mathbf{X}_i^{(\mathrm{out})}$ as logits (same as the right hand side of (2.18)). Masked language modelling is used with natural languages (Devlin et al., 2019), images (Qi et al., 2020) or proteins (Brandes et al., 2022).

2.2.4 Classification with Transformers

Transformer networks are compatible with classification problems where the input is a sequence and the output is a label. For that, add an additional first token $\mathbf{X}_1^{(0)} \in \mathbb{R}^{d_{\mathrm{hid}}}$ to the input matrix $\mathbf{X}^{(0)}$. We treat this token as a trainable vector and refer to it as a *class token*. The rest of tokens $\mathbf{X}_i^{(0)}$, $i > 1$, are treated in the same way as before (2.19). As the result, the sequence length is increased by 1 ($L \to L+1$). After the Transformer layers are applied, the final class prediction is obtained from $\mathbf{X}_1^{(\mathrm{out})}$ as logits, i.e. d_{out} is the number of classes. Therefore, the sequence of tranformations $\mathbf{X}_1^{(0)} \to \cdots \to \mathbf{X}_1^{(s)} \to \mathbf{X}_1^{(\mathrm{out})}$ can be thought as a composition of attention lookups at different levels of the Transformer backbone. The minimized loss $\mathscr{L}$ is negative logarithm of the probability of the correct class.

2.2.5 Vision Transformers

Vision Transformers (ViT) (Dosovitskiy et al., 2021) were proposed as an alternative to convolutional neural networks (He et al., 2015) for image classification. The input image of size $H \times W$ is split into non-overlapping patches of size $H_\text{p} \times W_\text{p}$. Then each i'th patch is flattened into a vector $\mathbf{q}^{(i)} \in \mathbb{R}^{H_\text{p} W_\text{p}}$. The first token $\mathbf{X}_1^{(0)}$ is the class token as described in Section 2.2.4. All other tokens at positions $2 \leq i \leq L$ are defined as follows:

$$\forall 2 \leq i \leq L : \mathbf{X}_i^{(0)} = \mathbf{W}_\text{p} \mathbf{q}^{(i-1)} + \mathbf{posemb}_i$$

where $\mathbf{W}_\text{p} \in \mathbb{R}^{d_\text{hid} \times (H_\text{p} W_\text{p})}$ is a trainable matrix. After Transformer layers are applied, the final class prediction is obtained from $\mathbf{X}_1^{(out)}$ as logits as described in Section 2.2.4.

2.2.6 Literature overview: Transformer networks in deep learning

Natural language processing. Initially proposed in the context of neural machine translation (Vaswani et al., 2017), Transformers were later adapted to state-of-the-art solutions in many other natural language processing tasks such as dialog systems (Zhang et al., 2020b), named entity recognition (Yamada et al., 2020), speech recognition (Hsu et al., 2021; Luo et al., 2020), text summarization (Liu and Lapata, 2019), part-of-speech tagging (Heinzerling and Strube, 2019), question answering (Liu et al., 2019), sentiment analysis and text classification (Yang et al., 2019) and many other tasks. Perhaps, one of the most prominent recent directions in natural language processing is language modelling. Large language models with billions of parameters trained on massive Internet data were shown to have remarkable generalization capabilities in the form of a few-shot adaptation to new tasks (Brown et al., 2020; Radford et al., 2019). The recent trend was to increase the number of parameters in Transformer-based language models from millions (Radford et al., 2018) to tens (Shoeybi et al., 2019) and hundreds (Brown et al., 2020) of billions and finally to trillions (Fedus et al., 2022) of parameters. Another popular direction for improving generalization is pretraining via masked language modelling on large text corpora (Devlin et al., 2019).

Computer vision. After numerous successes in natural language processing, Transformers have been also adapted in the computer vision domain. Transformers are used for image segmentation (Gong et al., 2021a), object detection (Carion et al., 2020; Zhu et al., 2021), image generation (Chen et al., 2021), image super-resolution (Yang et al., 2020), visual question answering (Tan and Bansal, 2019), video understanding (Girdhar et al., 2019; Sun et al., 2019), point cloud classification and segmentation (Guo et al., 2020; Zhao et al., 2021). Inspired by the success of pretrained Transformers in natural language processing,

Dosovitskiy et al. (2021) proposed Vision Transformer which is pretrained on a massive set of images and later fine-tuned on a small or moderate-size image recognition dataset. Similar technique was adapted for video (Arnab et al., 2021) and audio recognition (Gong et al., 2021b). Internet-scale Transformer pretraining was involved in creating two-tower (Radford et al., 2021) and generative models (Ramesh et al., 2021) for image-text mappings.

Bioinformatics. In biology, Transformers were adapted for autoregressive protein modelling where the obtained language models are capable to produce new proteins with stable chemical properties (Madani et al., 2020) and to predict structure and function of the given protein without an expensive physical simulation (Elnaggar et al., 2019; Rives et al., 2019). An alternative to language modelling of proteins is energy-based models (Du et al., 2020). Also, Transformer can be applied directly on the spatial representation of the protein (Ingraham et al., 2019).

Chapter 3

Performer: random features for attention approximation

3.1 Motivation

Transformers rely on a trainable self-attention mechanism that identifies complex dependencies between the elements of each input sequence (e.g. amino acids within a protein). Unfortunately, a standard Transformer scales quadratically ($O(L^2d)$, Section 2.2.2) with the number of tokens L in the input sequence, which is prohibitively expensive for large L. Several solutions have been proposed to address this issue (Bello et al., 2019; Beltagy et al., 2020; Chan et al., 2020; Child et al., 2019; Gulati et al., 2020). Most approaches restrict the attention mechanism to attend to local neighborhoods (Parmar et al., 2018) or incorporate structural priors on attention such as sparsity (Child et al., 2019), pooling-based compression (Rae et al., 2020) clustering/binning/convolution techniques (e.g. (Roy et al., 2020) which applies k-means clustering to learn dynamic sparse attention regions, or (Kitaev et al., 2020), where locality sensitive hashing is used to group together tokens of similar embeddings), sliding windows (Beltagy et al., 2020), or truncated targeting (Chelba et al., 2020). Thus these approaches do not aim to approximate regular attention, but rather propose simpler and more tractable attention mechanisms, often by incorporating additional constraints (e.g. identical query and key sets as in (Kitaev et al., 2020)), or by trading regular attention with sparse attention using more layers (Child et al., 2019). Unfortunately, there is a lack of rigorous guarantees for the representation power produced by such methods, and sometimes the validity of sparsity patterns can only be verified empirically through trial and error. Other techniques which aim to improve the computational complexity of Transformers include reversible residual layers allowing for one-time activation storage in training (Kitaev et al.,

2020) and shared attention weights (Xiao et al., 2019). These constraints may impede application to problems that involve long sequences, where approximations of the self-attention mechanism are not sufficient. Approximations based on truncated back-propagation (Dai et al., 2019) where the gradient is passed only through a portion of the sequence are also unable to capture long-distance correlations since the gradients are only propagated inside a localized window.

Recent work has demonstrated that Transformers fit to the amino acid sequences of single proteins learn to accurately predict information about protein structure and function, and can generate new sequences with specific properties (Elnaggar et al., 2019; Madani et al., 2020; Rives et al., 2019). Approaches that encode 3D protein structural data via Transformer-based models demonstrate improved performance, despite the restriction of attention to the local structural neighborhoods of each node (Du et al., 2020; Ingraham et al., 2019). These models provide initial promise for protein design applications, but their applicability beyond the design of single proteins is limited because they truncate sequences to 512 or 1024 amino acids. The ability to scale to longer sequences without imposing sparsity constraints would enable the use of Transformers to jointly model multiple concatenated protein sequences and the interactions between them. This follows recent works employing simpler statistical models that predict protein structure, protein-protein interactions and protein interaction networks from evolutionary sequence data (Bitbol et al., 2016; Cong et al., 2019; Hopf et al., 2012; Ovchinnikov et al., 2014; Weigt et al., 2009).

In this chapter, we present a new Transformer architecture, *Performer*[1], based on *Fast Attention Via Orthogonal Random features* (FAVOR). Our proposed mechanism has several advantageous properties: it scales linearly rather than quadratically in the length of the sequence and it is characterized by sub-quadratic space complexity. Furthermore, it provides strong theoretical guarantees: unbiased estimation of the regular attention matrix and uniform convergence. FAVOR is designed for long input sequences where the sequence length L satisfies $L \gg d$ where d is the query dimension. In contrast to previous approaches, instead of simplifying regular attention via various structural priors (which can lead to different, potentially incompatible architectures), we show that it can be effectively approximated as it is, without any "liftings". This leads to our method being flexible: combined with small amounts of fine-tuning, the Performer is backwards-compatible with pretrained regular Transformers and can be also used beyond the Transformer scope as a more scalable replacement for regular attention. We demonstrate its effectiveness on challenging tasks that include protein sequence modeling and autoregressive image generation on ImageNet (Deng et al., 2009).

[1]Implementation: https://github.com/google-research/google-research/tree/master/performer/fast_attention

We show that regular attention can be considered a special case of a much larger class of kernel-driven attention mechanisms, generalized attention, and that all our results for regular attention can be directly translated also to this extended class. This observation enables us to explore a much larger class of attention models.

The structure of the chapter is as follows:

- Section 3.2 describes the main idea and gives mathematical details.
- Section 3.3 describes the final algorithm for the efficient causal and noncausal self-attention.
- Section 3.4 analyses computational complexity of the algorithm.
- Section 3.5 provides theoretical guarantees regarding the uniform concentration of the FAVOR mechanism.
- Section 3.6 presents a general class of kernel-based attention mechanisms, generalized attention, which encapsulates FAVOR as a special case.
- Section 3.8 presents an extensive empirical evaluation of FAVOR and generalized attention in a number of tasks from natural language processing, computer vision and protein modelling.
- Section 3.9 is reserved for the discussion.

3.2 FAVOR: approximating self-attention with random features

We start with the case of noncausal self-attention. Recall its definition from Chapter 2 (2.16):

$$\mathrm{Att}_{\leftrightarrow}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{D}^{-1}\mathbf{A}\mathbf{V} \in \mathbb{R}^{L\times d}, \quad \mathbf{A} = \exp[\mathbf{Q}\mathbf{K}^{\top}/\sqrt{d}], \quad \mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{1}_L).$$

where $\mathbf{Q},\mathbf{K},\mathbf{V} \in \mathbb{R}^{L\times d}$, L is the sequence dimension of the Transformer and d is the query dimension. What we observe is that, essentially, computing noncausal self-attention reduces to applying $\mathbf{A}$ as a linear map to the matrix $\begin{bmatrix}\mathbf{V} & \mathbf{1}_L\end{bmatrix}$.

Instead of computing and storing the attention matrix $\mathbf{A} \in \mathbb{R}^{L\times L}$ explicitly, we derive its unbiased stochastic approximation, which benefits from low-rank structure.

Take some $1 \leq i, j \leq L$. The (i, j)-th element of $\mathbf{A}$ can be expressed as:

$$\mathbf{A}_{i,j} = \exp\left(\frac{\mathbf{Q}_i^\top \mathbf{K}_j}{\sqrt{d}}\right) = \exp\left(\frac{\|\mathbf{Q}_i\|^2}{2\sqrt{d}}\right) \exp\left(-\frac{\|\mathbf{Q}_i - \mathbf{K}_j\|^2}{2\sqrt{d}}\right) \exp\left(\frac{\|\mathbf{K}_j\|^2}{2\sqrt{d}}\right). \tag{3.1}$$

In other words, the self-attention matrix $\mathbf{A}$ can be decomposed as:

$$\mathbf{A} = \mathbf{D_Q} \times \mathscr{K} \times \mathbf{D_K}, \quad \mathbf{D_Q} = \mathrm{diag}\Big(\exp(\|\mathbf{Q}_1\|^2/2\sqrt{d}), \ldots, \exp(\|\mathbf{Q}_L\|^2/2\sqrt{d})\Big), \tag{3.2}$$

$$\mathbf{D_K} = \mathrm{diag}\Big(\exp(\|\mathbf{K}_1\|^2/2\sqrt{d}), \ldots, \exp(\|\mathbf{K}_L\|^2/2\sqrt{d})\Big) \tag{3.3}$$

where $\mathscr{K}$ is the Gaussian kernel matrix evaluated on $\{\mathbf{x}^{(i)} = d^{-1/4}\mathbf{Q}_i \in \mathbb{R}^d\}_{1 \leq i \leq L}$ and $\{\mathbf{y}^{(j)} = d^{-1/4}\mathbf{K}_j \in \mathbb{R}^d\}_{1 \leq j \leq L}$. Both matrices $\mathbf{D_Q}$ and $\mathbf{D_K}$ are diagonal. They can be computed in $O(Ld)$ time and applied as linear maps efficiently.

Hence, we can again use the unbiased low-rank approximation of $\mathscr{K}$ using random features:

$$\mathscr{K} = \mathbb{E}\,\mathrm{Re}\left(\mathbf{S}^{(1)} \times (\mathbf{S}^{(2)})^\top\right)$$

where $\mathbf{S}^{(1)}, \mathbf{S}^{(2)}$ are random matrices defined in (2.7-2.8) for $\{\mathbf{x}^{(i)} = d^{-1/4}\mathbf{Q}_i \in \mathbb{R}^d\}_{1 \leq i \leq L}$ and $\{\mathbf{y}^{(j)} = d^{-1/4}\mathbf{K}_j \in \mathbb{R}^d\}_{1 \leq j \leq L}$.

Using this decomposition, we can derive an unbiased approximation of $\mathbf{AC}$, where $\mathbf{C} = \begin{bmatrix}\mathbf{V} & \mathbf{1}_L\end{bmatrix}$:

$$\begin{aligned}\mathbf{AC} &= \mathbb{E}\left(\mathbf{D_Q} \times \mathrm{Re}\left(\mathbf{S}^{(1)} \times (\mathbf{S}^{(2)})^\top\right) \times \mathbf{D_K}\right)\mathbf{C} = \mathbb{E}\left(\mathrm{Re}\left(\mathbf{D_Q}\mathbf{S}^{(1)} \times (\mathbf{D_K}\mathbf{S}^{(2)})^\top\right)\right)\mathbf{C} \\ &= \mathbb{E}\,\mathrm{Re}\left(\mathbf{P}^{(1)} \times (\mathbf{P}^{(2)})^\top \times \mathbf{C}\right) = \mathbb{E}\,\mathrm{Re}\left(\mathbf{P}^{(1)} \times ((\mathbf{P}^{(2)})^\top \times \mathbf{C})\right)\end{aligned}$$

where $\mathbf{P}^{(1)} = \mathbf{D_Q}\mathbf{S}^{(1)}, \mathbf{P}^{(2)} = \mathbf{D_K}\mathbf{S}^{(2)}$, $\mathbf{P}^{(1)}, \mathbf{P}^{(2)} \in \mathbb{C}^{L \times M}$, that is

$$\forall 1 \leq i \leq L, 1 \leq m \leq M : \mathbf{P}^{(1)}_{i,m} = M^{-1/2}\exp(\|\mathbf{Q}_i\|^2/2\sqrt{d}) f^{(1)}(\boldsymbol{\omega}^{(m)}, d^{-1/4}\mathbf{Q}_i), \tag{3.4}$$

$$\forall 1 \leq j \leq L', 1 \leq m \leq M : \mathbf{P}^{(2)}_{j,m} = M^{-1/2}\exp(\|\mathbf{K}_j\|^2/2\sqrt{d}) f^{(2)}(\boldsymbol{\omega}^{(m)}, d^{-1/4}\mathbf{K}_j). \tag{3.5}$$

and M is the number of random features chosen by the user so that it's much smaller than L ($M \ll L$).

In this chapter, we opt for existing random features for the Gaussian kernel – TrigRFs (Section 2.1.3) – as the choice for $f^{(1)}(\boldsymbol{\omega}, \mathbf{x}), f^{(2)}(\boldsymbol{\omega}, \mathbf{x})$. In the next chapters we will discuss potential problems with TrigRF choice and will propose new improved random feature variants. We use an orthogonal variant of TrigRFs (Section 2.1.4). This results in the self-

attention approximation mechanism which we refer to as *F*ast *A*ttention *V*ia *O*rthogonal *R*andom features (FAVOR).

Algorithm 3 Outline of the FAVOR mechanism (causal and noncausal) which is also applicable for GA (defined in Section 3.6).

Input: $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L \times d}$, $M \in \mathbb{N}$, onGA, onCausal, onFullPS – binary flags.
Output: Approximation to $\widehat{\mathrm{Att}}_{\rightarrow}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \in \mathbb{R}^{L \times d}$ if onCausal. $\widehat{\mathrm{Att}}_{\leftrightarrow}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \in \mathbb{R}^{L \times d}$ otherwise.
Sample $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(M)}$ according to Algorithm 2;
if onGA **then**
    Compute $\mathbf{P}^{(1)}, \mathbf{P}^{(2)}$ according to (3.4,3.5);
else
    Compute $\mathbf{P}^{(1)} := \mathbf{P}^{(1,\mathrm{GA})}, \mathbf{P}^{(2)} := \mathbf{P}^{(2,\mathrm{GA})}$ according to (3.9);
end if
Set $\mathbf{C} = \begin{bmatrix} \mathbf{V} & \mathbf{1}_L \end{bmatrix}$;
if onCausal **then**
    if onFullPS **then**
        Set **psInp** $:= (\mathbf{C}_i (\mathbf{P}_i^{(2)})^\top)_{i=1}^L \in \mathbb{C}^{L \times (d+1) \times M}$;
        Compute $\mathbf{R} := \mathrm{PS}(\mathbf{psInp}) \in \mathbb{C}^{L \times (d+1) \times M}$;
        Compute $\mathbf{Buf}^{(2)} := (\mathbf{R}_i \mathbf{P}_i^{(1)})_{i=1}^L \in \mathbb{C}^{L \times (d+1)}$;
    else
        Set $\mathbf{R}^{\mathrm{cur}} := \mathbf{0}_{(d+1) \times M}$;
        Set $\mathbf{Buf}^{(2)} := \mathbf{0}_{L \times (d+1)}$;
        for $i = 1$ **to** L **do**
            Update $\mathbf{R}^{\mathrm{cur}} += \mathbf{C}_i (\mathbf{P}_i^{(2)})^\top$;
            Compute $\mathbf{Buf}_i^{(2)} := \mathbf{R}^{\mathrm{cur}} \mathbf{P}_i^{(1)}$;
        end for
    end if
else
    Compute $\mathbf{Buf}^{(1)} := (\mathbf{P}^{(2)})^\top \times \mathbf{C} \in \mathbb{C}^{M \times (d+1)}$;
    Compute $\mathbf{Buf}^{(2)} := \mathbf{P}^{(1)} \times \mathbf{Buf}^{(1)} \in \mathbb{C}^{L \times (d+1)}$;
end if
if not onGA **then**
    Set $\mathbf{Buf}^{(2)} := \mathrm{Re}\left(\mathbf{Buf}^{(2)}\right)$;
end if
Return $\mathrm{diag}(\mathbf{Buf}_{:,d+1}^{(2)})^{-1} \mathbf{Buf}_{:,:d}^{(2)}$;

3.3 Causal FAVOR and the final algorithm

In the causal case, we recall the definition of self-attention again:

$$\mathrm{Att}_{\rightarrow}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{A}}\mathbf{V} \in \mathbb{R}^{L\times d}, \quad \widetilde{\mathbf{A}} = \mathrm{tril}(\mathbf{A}), \quad \widetilde{\mathbf{D}} = \mathrm{diag}(\widetilde{\mathbf{A}}\mathbf{1}_L)$$

where $\mathbf{A}$ is defined in the same way as for noncausal self-attention. From the definition we see that the computation reduces to applying the linear map $\widehat{\mathbf{A}} = \mathrm{tril}(\mathbf{A})$ efficiently. We again apply an unbiased random-feature approximation and substitute $\mathbf{A}$ by $\mathbf{P}^{(1)}(\mathbf{P}^{(2)})^\top$. This time our goal is to compute $\mathrm{tril}\left(\mathrm{Re}\left(\mathbf{P}^{(1)}(\mathbf{P}^{(2)})^\top\right)\right)\mathbf{C}$ in subquadratic time where $\mathbf{C} = \begin{bmatrix}\mathbf{V} & \mathbf{1}_L\end{bmatrix}$. In order to do so, observe that for all $1 \leq i \leq L$:

$$\begin{aligned}\left(\mathrm{tril}\left(\mathrm{Re}\left(\mathbf{P}^{(1)}(\mathbf{P}^{(2)})^\top\right)\right)\mathbf{C}\right)_i &= \sum_{j=1}^{i}\mathbf{C}_j\,\mathrm{Re}\left((\mathbf{P}^{(2)}_j)^\top\mathbf{P}^{(1)}_i\right) = \mathrm{Re}\left(\sum_{j=1}^{i}(\mathbf{C}_j(\mathbf{P}^{(2)}_j)^\top)\mathbf{P}^{(1)}_i\right)\\ &= \mathrm{Re}\left(\left(\sum_{j=1}^{i}\mathbf{C}_j(\mathbf{P}^{(2)}_j)^\top\right)\mathbf{P}^{(1)}_i\right) \qquad (3.6)\end{aligned}$$

where in the first transition we use the definition of a matrix product and $\mathrm{tril}(\cdot)$, in the second transition we use associativity of the matrix product and linearity of $\mathrm{Re}\,(\cdot)$ and in the third transition – distributive law of the matrix product with respect to the sum operation. The right hand side of (3.6) can be evaluated efficiently for all $1 \leq i \leq L$ by running a loop over $i = 1,\ldots,L$, updating the sum $\sum_{j=1}^{i}\mathbf{C}_j(\mathbf{P}^{(2)}_j)^\top$ and multiplying it by $\mathbf{P}^{(1)}_i$ inside every iteration.

An alternative way to evaluate (3.6) is to allocate a 3-dimensional tensor $(\mathbf{C}_i(\mathbf{P}^{(2)}_i)^\top)_{i=1}^{L} \in \mathbb{C}^{L\times(d+1)\times M}$. Then, compute its *prefix sum*

$$\mathbf{R} = \mathrm{PS}((\mathbf{C}_i(\mathbf{P}^{(2)}_i)^\top)_{i=1}^{L}) \qquad (3.7)$$

where $\mathrm{PS}(\cdot)$ computes prefix sum along the first dimension of the input tensor: for $\mathbf{Z} \in \mathbb{C}^{d_1\times\cdots\times d_t}$, $\mathrm{PS}(\mathbf{Z}) = (\sum_{j=1}^{i}\mathbf{Z}_j)_{i=1}^{d_1} \in \mathbb{C}^{d_1\times\cdots\times d_t}$. After that, according to the right hand side of (3.6), for all $1 \leq i \leq L$ we have

$$(\mathrm{tril}(\mathbf{P}^{(1)}(\mathbf{P}^{(2)})^\top)\mathbf{C})_i = \mathbf{R}_i\mathbf{P}^{(1)}_i.$$

Algorithm 3 summarizes the whole FAVOR computation for both causal and noncausal self-attention.[1] It accepts two binary flags: onCausal deciding which type of self-attention

[1]For brevity, Algorithm 3 also covers the case of generalized attention (GA) discussed later in Section 3.6. Whether to use FAVOR or GA is controlled by the binary flag onGA which we set to False for now.

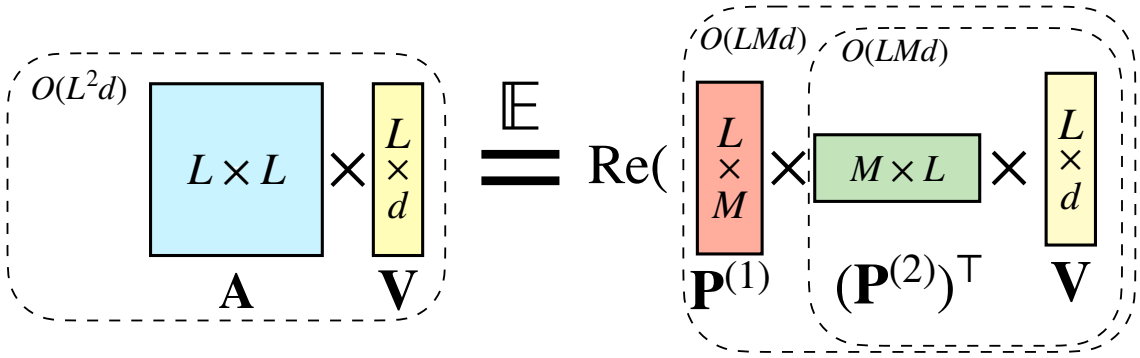


Fig. 3.1 Approximation of $\mathbf{A} \times \mathbf{V}$ by $\mathrm{Re}\left(\mathbf{P}^{(1)} \times (\mathbf{P}^{(2)})^\top \times \mathbf{V}\right)$ in noncausal FAVOR.

to approximate and onFullPS which is used with the causal variant to choose whether to compute the full 3-dimensional prefix sum tensor or use the iterative algorithm. We discuss the differences between two causal variant computations in the next section.

3.4 Complexity analysis

The complexity analysis of the noncausal FAVOR variant (Algorithm 3) is the same as for the standard random feature approximation (Section 2.1.2), i.e. the final computational complexity is $O(LMd)$ (since $L' = L$ and $n = d+1$ in the notation of Section 2.1.2). One needs to store $L \times M$-sized ($P^{(1)}$, $P^{(2)}$), $L \times (d+1)$-sized ($\mathbf{Buf}^{(2)}$) and $(d+1) \times M$-sized matrices ($\mathbf{Buf}^{(1)}$), resulting in $O(LM + Ld + Md)$ memory complexity.

The iterative causal variant requires $O(Md)$ computations inside each iteration over $i = 1, \ldots, L$, hence resulting in total $O(LMd)$ computational complexity. It also requires to store $L \times M$-sized ($P^{(1)}$, $P^{(2)}$), $L \times (d+1)$-sized ($\mathbf{Buf}^{(2)}$) and $(d+1) \times M$-sized matrices ($\mathbf{R}^{\mathrm{cur}}$), resulting in $O(LM + Ld + Md)$ memory complexity.

On the other hand, the full prefix sum version of the causal FAVOR requires $O(LMd)$ memory because of storing 3-dimensional tensors $\mathbf{psInp}, \mathbf{R} \in \mathbb{C}^{L \times (d+1) \times M}$. However, as opposed to the sequential loop, this version is parallelizable since there are $O(\log L)$ parallel time algorithms for computing prefix sum (Cormen et al., 2009; Ladner and Fischer, 1980). These algorithms still require a linear number of floating-point operations, hence giving $O(LMd)$ computational complexity but $O(\log L)$ parallel time complexity.

Figures 3.1 and 3.2 illustrate algorithms for causal and noncausal FAVOR respectively.

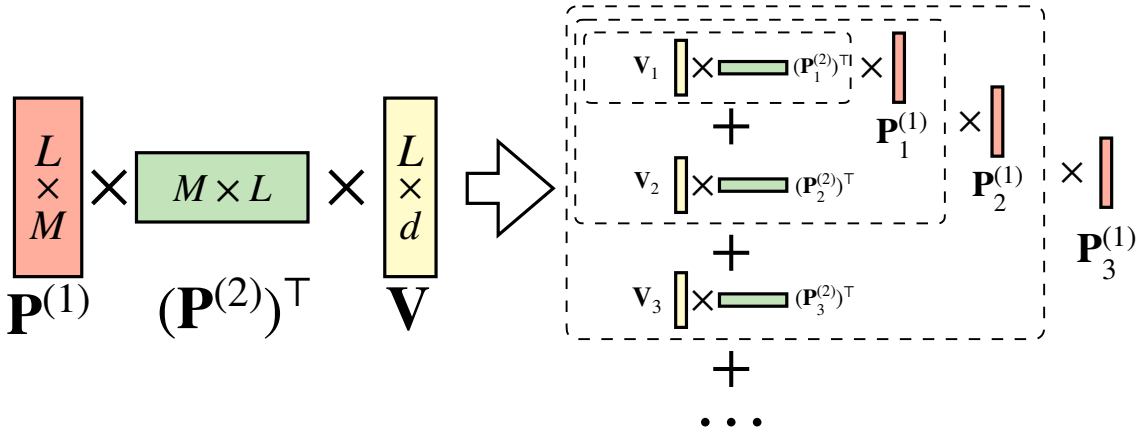


Fig. 3.2 What changes in Figure 3.1 in the case of noncausal FAVOR: matrix multiplications are replaced with prefix sums.

3.5 Concentration analysis

In contrast to other methods approximating the self-attention matrix $\mathbf{A}$, FAVOR provides provable uniform convergence guarantees for compact domains. We show that M_{opt}, the optimal number of random features, does not depend on L but only on d. In fact, we prove that if we take $M_{\text{opt}} = O(d\log(d))$, then with $O(Ld^2\log(d))$ time, we can approximate $\mathbf{A}$ up to any precision, regardless of the number of tokens L. In order to provide those guarantees for FAVOR, we leverage recent research on the theory of negative dependence for ORFs (Lin et al., 2020). The following is true:

Theorem 4. *Assume that the L_2-norm of rows of matrices $\mathbf{Q},\mathbf{K} \in \mathbb{R}^{L\times d}$ is upper-bounded by $R > 0$. Assume that $\widehat{\mathbf{A}}$ is an approximation of $\mathbf{A}$ from (2.16) obtained using M TrigRFs with i.i.d. or block-orthogonal $\boldsymbol{\omega}^{(m)}$'s. Let $\varepsilon > 0$. The minimum M, which is enough for $\|\widehat{\mathbf{A}} - \mathbf{A}\|_\infty \leq \varepsilon$ with any constant probability, satisfies*

$$M = O\left(d\varepsilon^{-2}\exp(d^{-1/2}R^2)\left(\log(4R\varepsilon^{-1}d^{3/4}) + d^{-1/2}R^2\right)\right).$$

According to the theorem, assuming that $d^{-1/2}R$ is constant (for the numerical stability of self-attention during training), the number of random features grows as $O(d\log d)$ to reach the ε-level of approximation and, most importantly, does not depend on L.

3.6 Generalized attention and Performer

The random feature interpretation of self-attention creates a new perspective of looking at it and extending it to new attention mechanisms which could potentially result in better models. *Generalized attention* (GA) is an idea to use, instead of batteries of random features $f^{(1)}(\boldsymbol{\omega}^{(m)},\cdot), f^{(2)}(\boldsymbol{\omega}^{(m)},\cdot)$ for the Gaussian kernel, an arbitrary user-defined mapping $g:\mathbb{R}^d \to \mathbb{R}^M$ where $M \ll L$. In this case, noncausal self-attention is defined as

$$\mathrm{Att}^{\mathrm{GA}}_{\leftrightarrow}(\mathbf{Q},\mathbf{K},\mathbf{V}) = (\mathbf{D}^{\mathrm{GA}})^{-1}\mathbf{A}^{\mathrm{GA}}\mathbf{V} \in \mathbb{R}^{L\times d},$$
$$\mathbf{A}^{\mathrm{GA}} = (g(\mathbf{Q}_i)^\top g(\mathbf{K}_j))_{i,j=1}^{L,L}, \quad \mathbf{D}^{\mathrm{GA}} = \mathrm{diag}(\mathbf{A}^{\mathrm{GA}}\mathbf{1}_L).$$

The noncausal variant, on the other hand, is defined as

$$\mathrm{Att}^{\mathrm{GA}}_{\rightarrow}(\mathbf{Q},\mathbf{K},\mathbf{V}) = (\widetilde{\mathbf{D}}^{\mathrm{GA}})^{-1}\widetilde{\mathbf{A}}^{\mathrm{GA}}\mathbf{V} \in \mathbb{R}^{L\times d}, \quad \widetilde{\mathbf{A}}^{\mathrm{GA}} = \mathrm{tril}(\mathbf{A}^{\mathrm{GA}}),$$
$$\widetilde{\mathbf{D}}^{\mathrm{GA}} = \mathrm{diag}(\widetilde{\mathbf{A}}^{\mathrm{GA}}\mathbf{1}_L). \tag{3.8}$$

As we see from this definition, the weight of a query-key pair in the differentiable dictionary is defined by the value of $g(\mathbf{Q}_i)^\top g(\mathbf{K}_j)$. The matrix $\mathbf{A}^{\mathrm{GA}} \in \mathbb{R}^{L\times L}$ can be alternatively expressed as

$$\mathbf{A}^{\mathrm{GA}} = \mathbf{P}^{(\mathrm{GA},1)} \times (\mathbf{P}^{(\mathrm{GA},1)})^\top,$$
$$\mathbf{P}^{(\mathrm{GA},1)} = (g_l(\mathbf{Q}_i))_{i,l=1}^{L,M} \in \mathbb{R}^{L\times M}, \quad \mathbf{P}^{(\mathrm{GA},2)} = (g_l(\mathbf{K}_j))_{j,l=1}^{L,M} \in \mathbb{R}^{L\times M}. \tag{3.9}$$

Because of this, we can use Algorithm 3 for computing $\mathrm{Att}^{\mathrm{GA}}_{\leftrightarrow}$ and $\mathrm{Att}^{\mathrm{GA}}_{\rightarrow}$ efficiently in time $O(LMd)$ instead of $O(L^2 d)$. Just substitute $\mathbf{P}^{(1)} \to \mathbf{P}^{(\mathrm{GA},1)}$, $\mathbf{P}^{(2)} \to \mathbf{P}^{(\mathrm{GA},2)}$ (i.e. set onGA = True).

Since g is a real-valued mapping, all computations in Algorithm 3 are real-valued and, therefore, the line 28 can be excluded since $\mathbf{Buf}^{(2)}$ is a real-valued matrix anyway.

We refer to a Transformer architecture (Section 2.2) where self-attention is replaced either by FAVOR or GA as *Performer*. In practice, g can be defined as $g(\mathbf{x}) = \mathrm{map}[\boldsymbol{\Omega}^\top \mathbf{x}]$ where $\boldsymbol{\Omega}$ can defined as a random $d \times M$ matrix with i.i.d. entries sampled from $\mathcal{N}(0,1)$, or as a random block-orthogonal matrix defined by Algorithm 2, or as an identity matrix $\boldsymbol{\Omega} = \mathbf{I}_d$ when $M = d$. map denotes an elementwise mapping which can be defined as, e.g. $\mathrm{map}(z) = \mathrm{ReLU}(z)$, or $\mathrm{ELU}(z)+1$, or z^2. We refer to such models as *Performer-ReLU*, *Performer-ELU* and *Performer-SQR* respectively.

3.7 Related work: other efficient Transformers

Here we discuss some other related methods for efficient self-attention approximation. We focus on the following three methods which we use as baselines in the experimental section of this and the following chapters.

Reformer (Kitaev et al., 2020) Reformer relies on the sparse approximation of the matrix $\mathbf{A}$ from (2.16). The goal is to approximate $\mathbf{A}$ with a sparse matrix $\mathbf{A}^{\text{Ref}}$ where only a small number of biggest entries of $\mathbf{A}$ are retained and all other small values are zeroed out. Since $\mathbf{A}_{i,j} = \exp(\mathbf{Q}_i^\top \mathbf{K}_j/\sqrt{d})$, the biggest entries of $\mathbf{A}$ correspond to pairs of $\mathbf{Q}_i$ and $\mathbf{K}_j$ with highest dot product value. For an approximate but efficient search of highest dot product pairs, authors of (Kitaev et al., 2020) apply *locality-sensitive hashing* (LSH). To apply this technique, additional restrictions are imposed on $\mathbf{Q}$ and $\mathbf{K}$ matrices:

- $\mathbf{K}$ is set to be equal to $\mathbf{Q}$. This is achieved by setting $\mathbf{W}_{\text{Q}}^{(l)} = \mathbf{W}_{\text{K}}^{(l)}$ in (2.15).
- L_2-norm of $\mathbf{Q}_i$ is set to 1 for all $1 \leq i \leq L$. This is achieved by normalizing $\mathbf{Q}_i$: $\mathbf{Q}_i^{\text{new}} = \mathbf{Q}_i/\|\mathbf{Q}_i\|$.

After this, locality-sensitive hash with $2b$ bins is computed by sampling a matrix $\mathbf{G}$ of size $d \times b$ with i.i.d. Gaussian entries from $\mathcal{N}(0,1)$. Then, the hash $\mathbf{h}_i$ for $\mathbf{Q}_i$ is computed as

$$\mathbf{h}_i = \text{argmax}\left(\begin{bmatrix}\mathbf{G} & -\mathbf{G}\end{bmatrix}^\top \times \mathbf{Q}_i\right)$$

where $\text{argmax}(\cdot)$ returns an index with the biggest entry of the input vector. Next, the pairs of $1 \leq i, j \leq L$ are selected where $\mathbf{h}_i = \mathbf{h}_j$, i.e. $\mathbf{Q}_i$ and $\mathbf{K}_j = \mathbf{Q}_j$ fall into the same hash bin. Such (i,j)-entries of $\mathbf{A}$ are retained in $\mathbf{A}^{\text{Ref}}$ and all other entries are zeroed out. This way, $\mathbf{A}^{\text{Ref}}$ is a permuted block-diagonal matrix. The hashing procedure can be repeated for several rounds and the sets of selected (i,j) can be intersected. The amount of nonzero entries of $\mathbf{A}^{\text{Ref}}$ can be controlled by changing b and the number of rounds. The complexity of applying sparse linear map $\mathbf{A}^{\text{Ref}}$ is proportional to the number of nonzero values in it, therefore, the smaller is the number of nonzero values, the faster is the approximation.

Linformer (Wang et al., 2020) Linformer, instead of approximating the linear map $\mathbf{A}$, uses random projections to compress $\mathbf{A}$ into shape $L \times k$ where $k < L$ and $\mathbf{V}$ into shape $k \times d$. More formally, the self-attention operation in Linformer is defined as follows:

$$\text{Att}_{\leftrightarrow}^{\text{Lin}}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{D}^{-1}\mathbf{A}^{\text{Lin}}\mathbf{E}\mathbf{V} \in \mathbb{R}^{L\times d}, \quad \mathbf{A}^{\text{Lin}} = \exp[\mathbf{Q}(\mathbf{E}\mathbf{K})^\top/\sqrt{d}] \in \mathbb{R}^{L\times K},$$

$$\mathbf{D} = \mathrm{diag}(\mathbf{A}^{\mathrm{Lin}}\mathbf{1}_K).$$

where $\mathbf{E}$ is a random projection matrix of size $k \times L$ with i.i.d. entries sampled from $\mathcal{N}(0, 1/k)$. This way, computational complexity is reduced from $O(L^2 d)$ to $O(Lkd)$ which is an improvement if k is much smaller than L.

Linear Transformer (Katharopoulos et al., 2020) Linear Transformer uses a similar idea of generic dot-product kernels to our generalized attention from Section 3.6. It can be thought as a special case of GA when $\mathrm{map}(z) = \mathrm{ELU}(z) + 1$ (i.e. Performer-ELU), $M = d$ and $\mathbf{\Omega}$ is an identity matrix.

3.8 Experiments

We implement our setup on top of pre-existing Transformer training code in Jax (Frostig et al., 2018) optimized with just-in-time (`jax.jit`) compilation, and complement our theory with empirical evidence to demonstrate FAVOR's practicality compared to the vanilla Transformer and other efficient Transformer models. Unless explicitly stated, the Performer replaces only the attention component with FAVOR, while all other components are exactly the same as for the vanilla Transformer. Furthermore, since we use negative log-likelihood loss in our generative modelling experiments, we report the standard accuracy metric (i.e. a fraction of correctly predicted elements of the sequence).

3.8.1 Computation costs

In this subsection, we empirically measure computational costs in terms of wall clock time T of the forward and backward pass. We experiment with the vanilla Transformer and Performer in two configurations: "Regular" ($(h, s, d_{\mathrm{ff}}, d_{\mathrm{hid}}) = (8, 6, 2048, 512)$) and "Small" ($(h, s, d_{\mathrm{ff}}, d_{\mathrm{hid}}) = (1, 6, 64, 64)$). We use a single V100 graphics processing unit (GPU) with 16 gigabyte memory in this experiment and noncausal FAVOR in both architectures. We measure scalability of both models in the following three scenarios:

1. Self-attention time complexities when comparing standard self-attention from Transformer and FAVOR from Performer (see Figure 3.3). We observe that FAVOR scales much better with a longer sequence length and also can handle longer sequences because of a smaller memory footprint.

2. Performer with a varying number of layers (see Figure 3.4). We observe that Performer can scale up to (but not necessarily limited to) 20 layers and that the dependence of wall clock time on L is linear for any number of layers.

3. Wall clock time complexities when comparing the vanilla Transformer and Performer models (see Figure 3.5). "X" (OPT) denotes the maximum possible speedup achievable, when attention simply returns the $\mathbf{V}$ matrix, showing that Performer is nearly optimal. We see that the maximum possible power of 2 length allowed is $2^{15} = 32768$ using the vanilla Transformer. We observe that in terms of L, Performer reaches nearly linear time complexity as opposed to the Transformer's quadratic time complexity. The Performer's memory consumption is also sub-quadratic (as it does not store the explicit $O(L^2)$-sized attention matrix), allowing higher batch sizes and longer sequence lengths. In fact, the Performer achieves nearly optimal speedup and memory efficiency possible, depicted by the "X"-line when attention is replaced by an "identity function" by simply returning the $\mathbf{V}$ matrix. The combination of both memory and backward pass efficiencies for large L has profound implications for training speed, as it allows, respectively, large batch training and lower wall clock time per gradient step, contributing to total train time reduction.

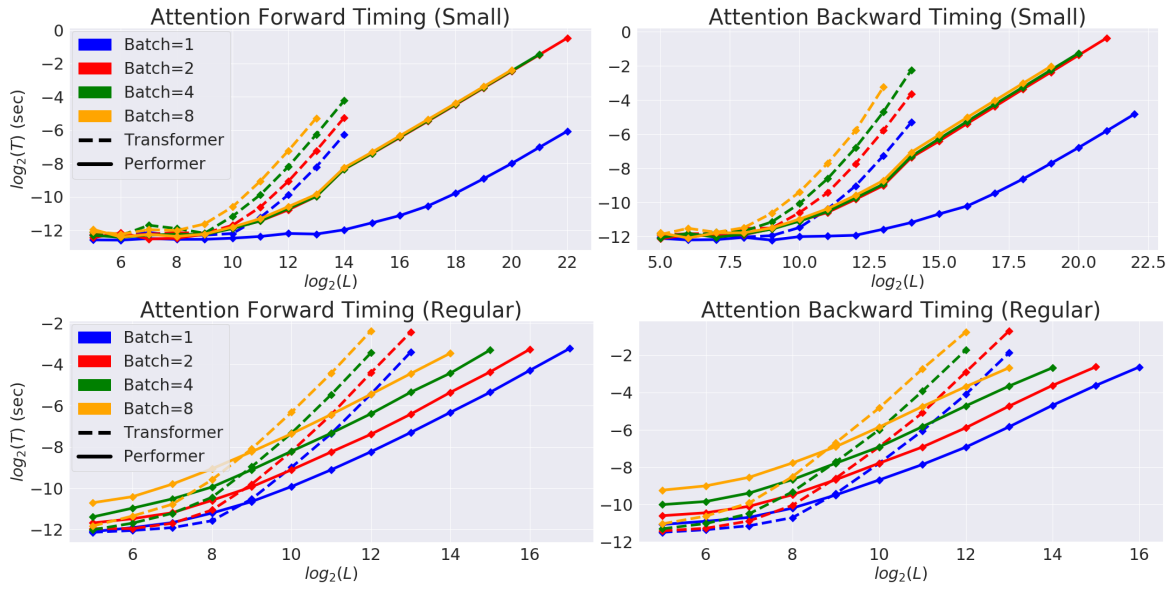


Fig. 3.3 Attention time complexities when comparing standard self-attention from Transformer and FAVOR from Performer. "Batch" in the legend stands for the batch size.

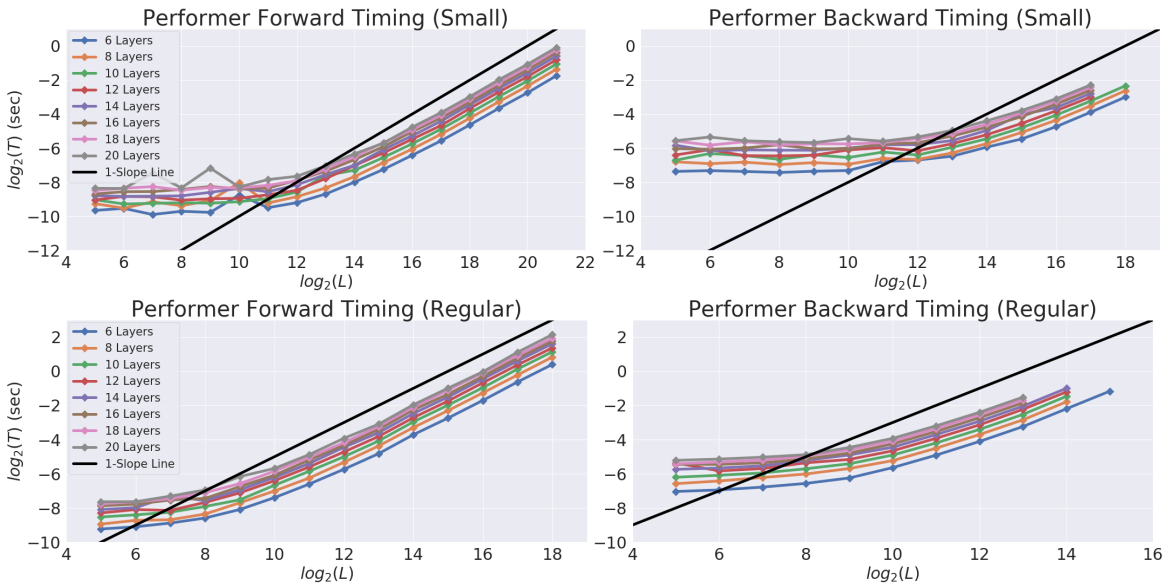


Fig. 3.4 Varying layers when using Performer. We show that our method can scale up to (but not necessarily limited to) even 20 layers.

3.8.2 Approximation error and compatibility with the vanilla Transformer

We further examine the approximation error of the full self-attention matrix in FAVOR. This approximation error directly affects the accuracy of FAVOR's output.

For Figure 3.6, we took $L = 4096, d = 16$, varied the number of random features M and measured the mean squared error (MSE) of self-attention matrix approximation. The figure demonstrates that orthogonal features generally produce lower error than unstructured i.i.d. features, which is in line with the theoretical results (Theorem 3).

In Figure 3.7, we demonstrate FAVOR approximation error propagation through the whole Transformer backbone. Since the error increases with the number of layers, we conclude that an additional fine-tuning is required after plugging FAVOR mechanism instead of the standard self-attention into a pretrained Transformer.

To validate this hypothesis, we conduct an additional experiment where we fine-tune the pretrained BERT (Devlin et al., 2019) model on masked language modelling for LM1B dataset (Chelba et al., 2014) (see Figure 3.8). We first fine-tune BERT with standard self-attention for 60000 steps, and then substitute self-attention by FAVOR and fine-tune for additional 60000 steps. We observe that FAVOR manages to show the same and, eventually, even a slightly higher performance compared to the pretrained baseline.

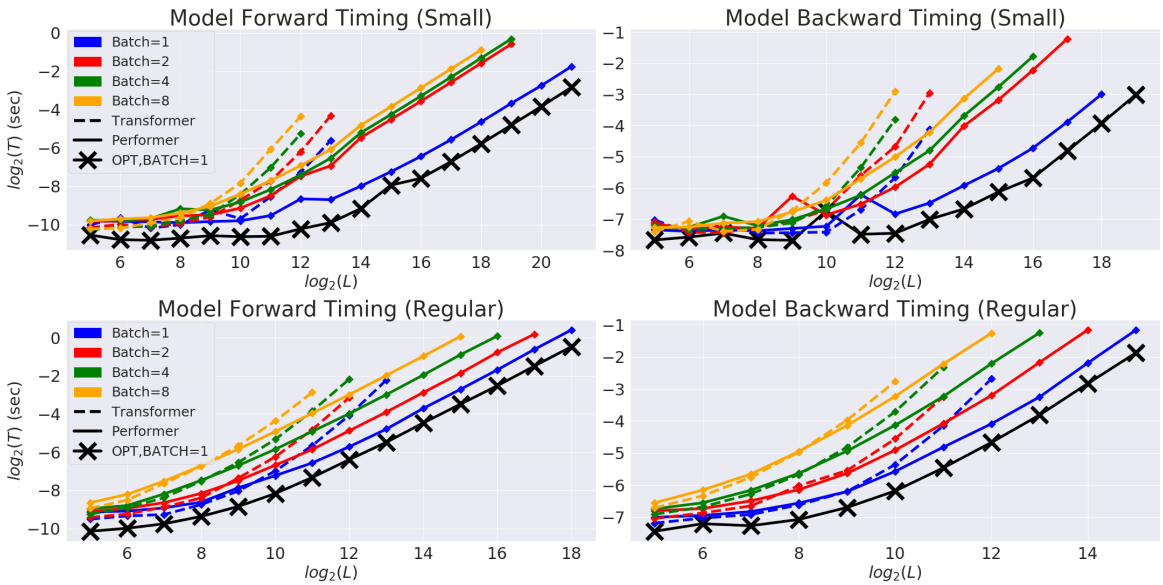


Fig. 3.5 Time complexities when comparing the Transformer and Performer models. "Batch" in the legend stands for the batch size.

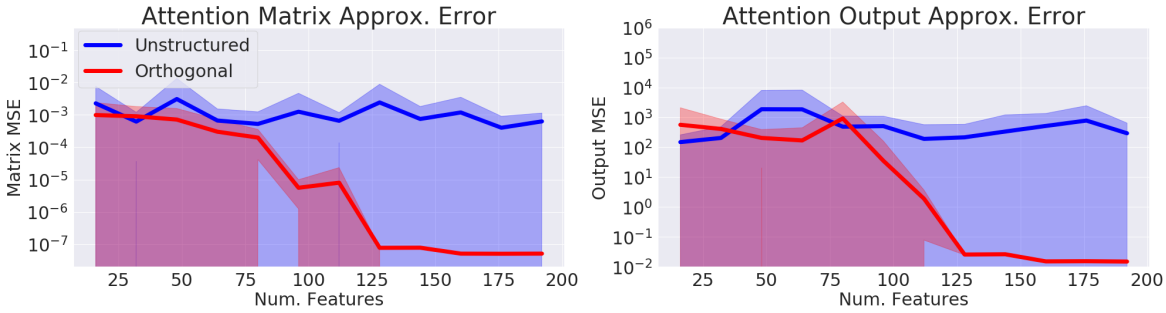


Fig. 3.6 Approximation errors for both the attention matrix and output of the mechanism itself. Standard deviations shown across 10 samples.

3.8.3 Multiple layer training

We further benchmark the Performer on both unidirectional (U) and bidirectional (B) FAVOR by training a 36-layer model for language modelling and masked language modelling on proteins. We take protein sequences from the January 2019 release of TrEMBL dataset (Consortium, 2019), similar to (Madani et al., 2020). As a baseline, along with Transformer, we also used Reformer which is another efficient Transformer with sparse self-attention approximation via LSH (Section 3.7). We opt for the exact same model parameters $(h, s, d_{\text{ff}}, d_{\text{hid}}) = (8, 36, 1024, 512)$ as in (Madani et al., 2020) for all runs. We conduct all experiments on 256 tensor processing units (TPUs). Batch sizes were maximized for each separate run given the corresponding compute constraints. For all experiments with the

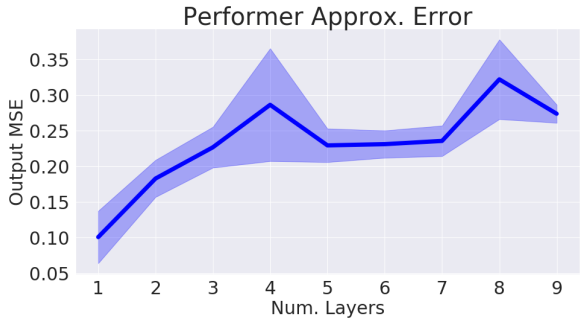


Fig. 3.7 Output approximation errors between the vanilla Transformer and Performer (with orthogonal features) for varying numbers of layers.

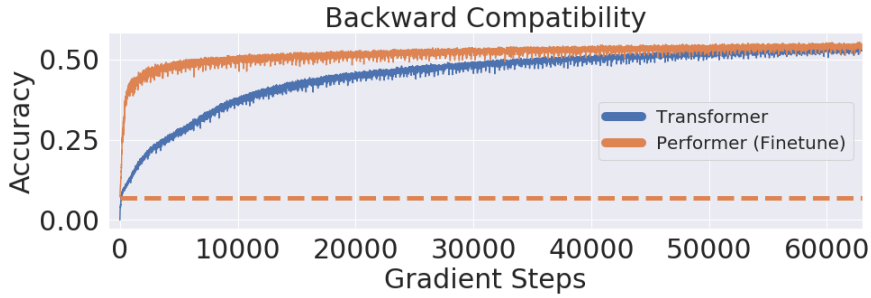


Fig. 3.8 Fine-tuning Performer from the parameters of vanilla Transformer on LM1B.

unidirectional (causal) FAVOR, we opt for onFullPS = False in Algorithm 3 which achieves good performance via Jax optimizations. Hyperparameters can be found in Appendix 3.B.

See Figure 3.9 for results. Reformer significantly drops in accuracy on the protein dataset. This suggests that sparse self-attention may be insufficient for protein tasks which require modelling of global interactions. Furthermore, the usefulness of generalized attention is evidenced by Performer-ReLU achieving the highest accuracy in both unidirectional and bidirectional cases. Our proposed self-attention approximation is also shown to be tight, achieving the same accuracies as the vanilla Transformer.

Table 3.1 contains tabular results for the protein modelling task. We report the following evaluation metrics:

1. **Accuracy**: For unidirectional models, we measure the accuracy on next-token prediction, averaged across all sequence positions in the dataset. For bidirectional models, we mask each token with 15% probability and measure accuracy across the masked positions.

2. **Perplexity**: For unidirectional models, we measure perplexity $2^{\mathscr{L}}$ where $\mathscr{L}$ is from (2.20). For bidirectional models, similar to the accuracy case, we measure masked perplexity ($2^{\mathscr{L}^{\mathrm{M}}}$) where $\mathscr{L}^{\mathrm{M}}$ is from (2.20).

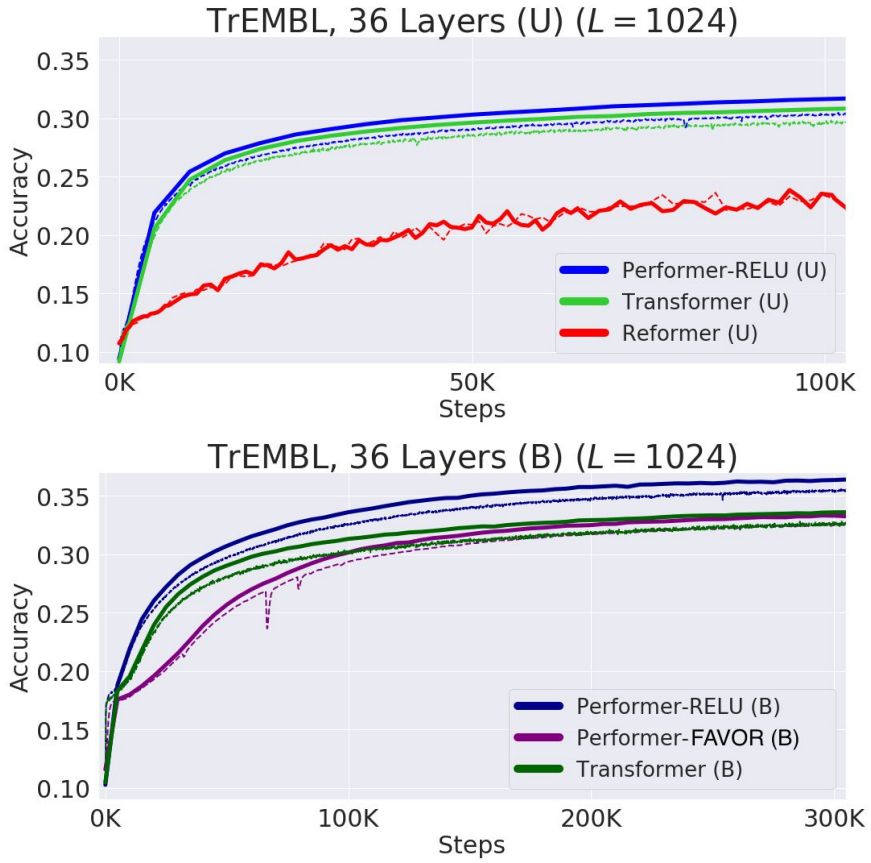


Fig. 3.9 TrEMBL protein modelling results. Train results correspond to dashed line, validation – to solid line. (U) stands for "unidirectional", (B) for "bidirectional".

Hyperparameters for the experiment can found in Appendix 3.B.

3.8.4 Large length training

We evaluate unidirectional (U) Performer on the task of language modelling on ImageNet (Deng et al., 2009) dataset where images are resized to 64×64 (referred to ImageNet64 (Parmar et al., 2018)), resulting in $L = 12288$ ($64 \times 64 \times 3$ because each of three channels is modelled separately). Note that, since this is an autoregressive modelling task where elements are modeled one after the other, we cannot use tokenized patches as discussed in Section 2.2.5. We took $(h, s, d_{\text{ff}}, d_{\text{hid}}) = (8, 6, 2048, 512)$ configurations for both the Performer and the Reformer, evaluated both at 50000 steps. We see that the Performer matches the Reformer, when both are trained on 64 TPUs. Depending on hardware (TPU or GPU), we also found that the Performer can be 2 times faster than the Reformer.

Model Type	Set Name	Model	Accuracy	Perplexity
UNI	Test	Empirical Baseline	9.92	17.80
		Transformer	30.80	9.37
		Performer-ReLU	31.58	9.17
	OOD	Empirical Baseline	9.07	17.93
		Transformer	19.70	13.20
		Performer-ReLU	18.44	13.63
BID	Test	Transformer	33.32	9.22
		Performer-ReLU	36.09	8.36
		Performer (FAVOR)	33.00	9.24
	OOD	Transformer	25.07	12.09
		Performer-ReLU	24.10	12.26
		Performer (FAVOR)	23.48	12.41

Table 3.1 TrEMBL protein modelling results (tabular). The empirical baseline results are applicable to both the unidirectional (UNI) and bidirectional (BID) models.

For a long-sequence protein language modelling study, we create an initial protein benchmark for modelling interactions among groups of proteins by concatenating protein sequences to length $L = 8192$ from TrEMBL, long enough to model protein interaction networks without the large sequence alignments required by existing methods (Cong et al., 2019). In this setting, a baseline Transformer overloads memory even at a batch size of 1 per chip, by a wide margin. Thus as a baseline we were forced to use a significantly smaller variant, reducing to $(h, s, d_{\text{ff}}, d_{\text{hid}}) = (8, \{1,2,3\}, 256, 256)$. Meanwhile, the Performer trains efficiently at a batch size of 8 per TPU chip using the standard $(8, 6, 2048, 512)$ architecture. We see in Figure 3.10 that the smaller Transformer ($n_{layer} = 3$) is quickly bounded at $\approx 19\%$ accuracy, while the Performer is able to train continuously to $\approx 24\%$ accuracy.

Hyperparameters for both experiments can found in Appendix 3.B.

3.8.5 Generalized attention

We investigate GA mechanisms introduced in Section 3.6 on TrEMBL when $L = 512$ for various kernel functions defined by elementwise mappings g. This is similar to (Tsai

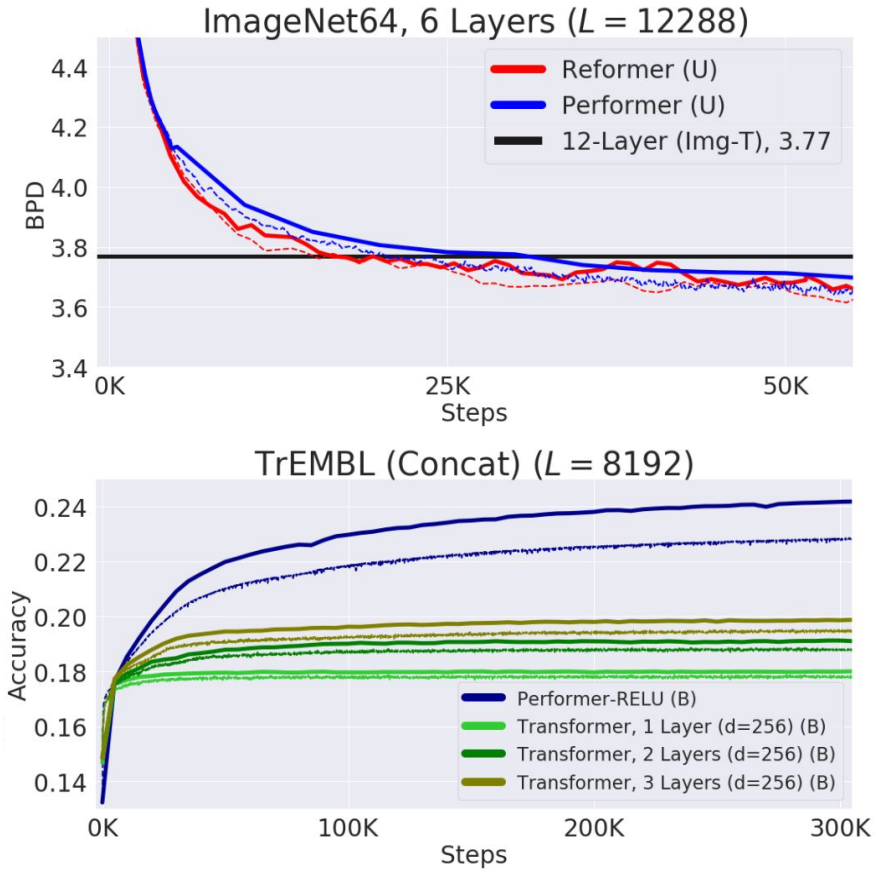


Fig. 3.10 Long-sequence evaluation of Performer. Img-T corresponds to the result of (Parmar et al., 2018) which uses 12-layers but with cropped lengths on a regular Transformer. (U) stands for "unidirectional", (B) for "bidirectional".

et al., 2019) which also experiment with various attention kernels for natural language processing. We compare different mappings "map" from the set of sigmoid $1/(1+\exp(-z))$, exp, ReLU, absolute value $|\cdot|$, GeLU, cos, tanh and identity mapping. Also, we compare block-orthogonal and i.i.d. $\mathbf{\Omega}$ (Figures 3.11 and 3.12). We use the batch size of 128 per device. We observe that map $=$ ReLU shows optimal performance with small deviation and orthogonal $\mathbf{\Omega}$ shows better average performance than i.i.d. variant.

Other hyperparameters are reported in Appendix 3.B.

3.8.6 Self-attention matrix illustration

In this section, we illustrate the self-attention matrices $\mathbf{D}^{-1}\mathbf{A}^{\mathrm{GA}}$ produced by Performer-ReLU. We focus on the bidirectional self-attention case and choose one Performer model trained on the single-protein language modelling on TrEMBL for 5×10^5 steps.

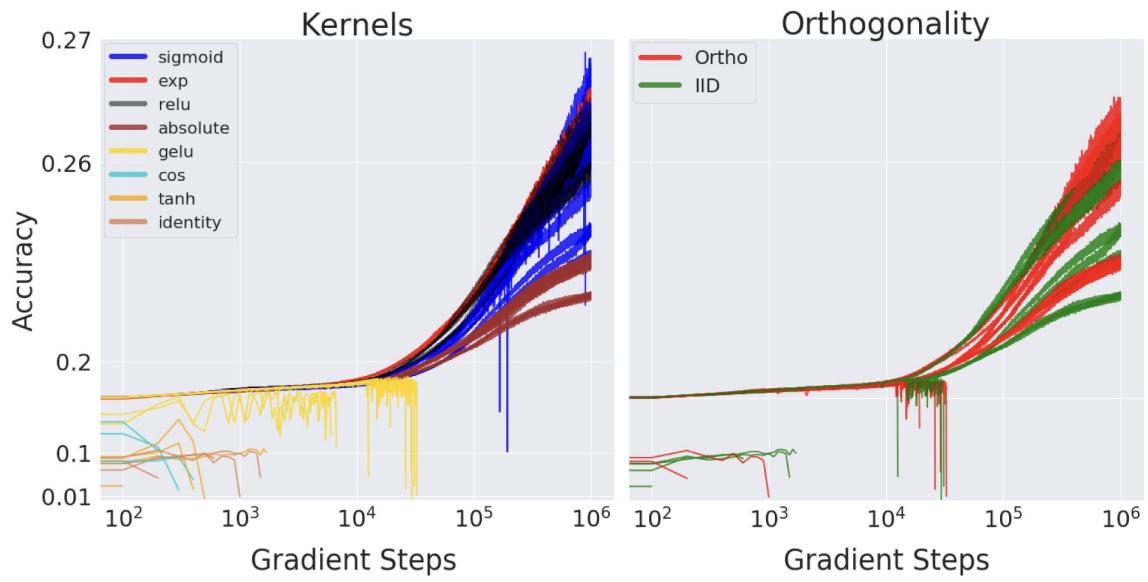


Fig. 3.11 GA comparison, different kernels highlighted (left) and orthogonal versus i.i.d. $\mathbf{\Omega}$ highlighted (right). The early stopping of the curve is due to NaN (not a number) errors in the model. All runs were performed on 4 TPUs.

We note that while the Transformer model instantiates the attention matrix $\mathbf{D}^{-1}\mathbf{A}$ in order to compute the attention output, the FAVOR mechanism returns the attention output directly (see Algorithm 3), so we perform an additional computation to find the matrix $\mathbf{D}^{-1}\mathbf{A}^{\mathrm{GA}}$ explicitly.

Self-attention matrix examples We start by visualizing the self-attention matrix for an individual protein sequence. We use the BPT1_BOVIN protein sequence[1], one of the most extensively studied globular proteins, which contains 100 amino acids. In Figure 3.13, we show the self-attention matrices for the first 4 layers. Note that many heads show a diagonal pattern, where each node attends to its neighbors, and some heads show a vertical pattern, where each head attends to the same fixed positions. These patterns are consistent with the patterns found in Transformer models trained on natural language (Kovaleva et al., 2019). In Figure 3.15 we highlight these attention patterns by focusing on the first 25 tokens, and in Figure 3.14, we illustrate in more detail two attention heads.

Amino acid similarity Furthermore, we analyze the amino-acid similarity matrix estimated from the self-attention matrices produced by the Performer model, as described in (Vig et al.,

[1]https://www.uniprot.org/uniprot/P00974

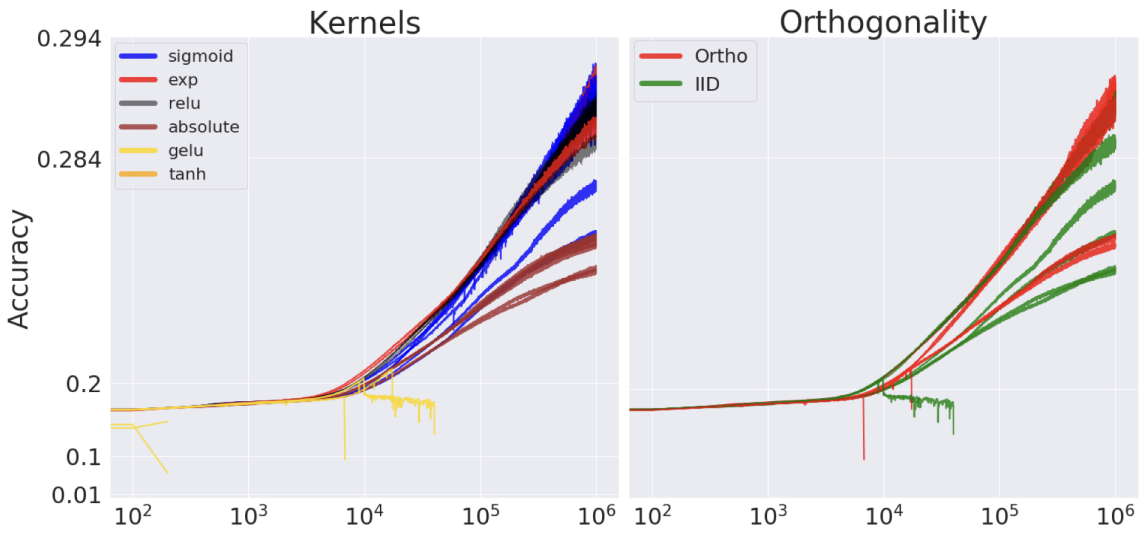


Fig. 3.12 The version of Figure 3.11 trained on 16 TPUs.

2020). We aggregate the self-attention matrix across 800 sequences. The resulting similarity matrix is illustrated in Figure 3.16. Note that the Performer recognises highly similar amino acid pairs such as (D, E) and (F, Y).

3.9 Discussion

In this chapter, we introduced *F*ast *A*ttention *V*ia *O*rthogonal *R*andom features (FAVOR): a mechanism for efficient approximation of the self-attention matrix working both in the causal and noncausal case. FAVOR's complexity is only $O(LMd)$ compared to exact computation's complexity of $O(L^2d)$, $M \ll L$ is a user-defined number of random features which controls the tradeoff between precision and computations. We further proposed FAVOR's extension – generalized attention (GA), a class of efficient self-attention mechanisms. We refer to the Transformer equipped with FAVOR or GA as Performer. We provided theoretical bounds of FAVOR approximation and extensively evaluated Performer in real-world applications such as image, text and protein modelling. In the subsequent chapters, we will discuss improvements of FAVOR and other computational properties of Performer.

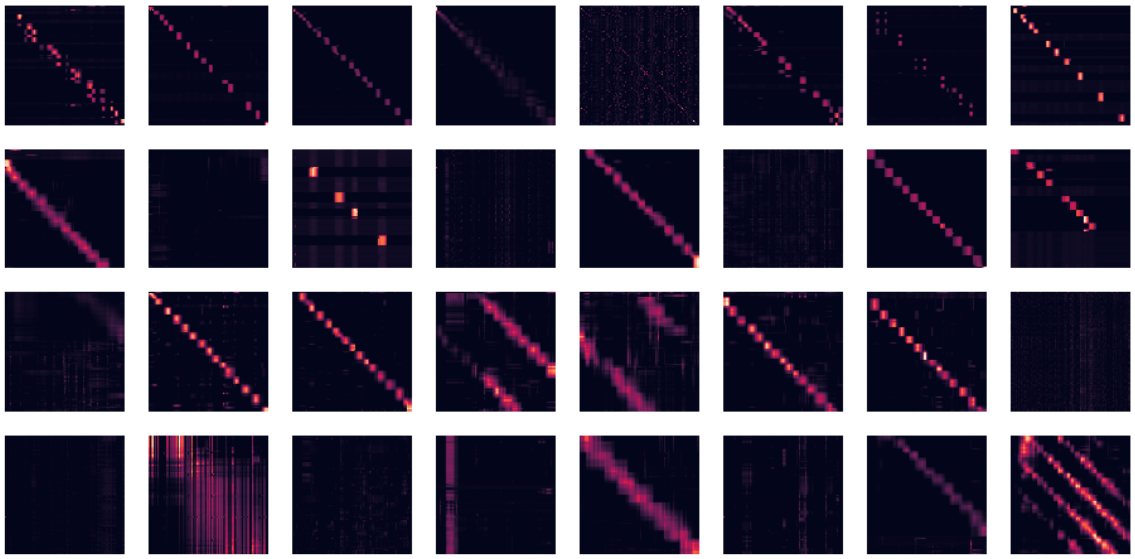


Fig. 3.13 The self-attention matrices (each row is a layer, each column is head index, each cell contains the self-attention matrix across the entire BPT1_BOVIN protein sequence).

Appendix 3.A Proofs

Proof of Theorem 4

Proof. We rely on Theorem 3 from (Lin et al., 2020). Note that we can apply it in our case, since $\mathrm{Re}\left(f^{(1)}(\boldsymbol{\omega},\mathbf{x})f^{(2)}(\boldsymbol{\omega},\mathbf{y})\right)=\cos(\boldsymbol{\omega}^\top(\mathbf{x}-\mathbf{y}))$, i.e. $f=\cos$ in the notation of Theorem 3 from (Lin et al., 2020), which is a bounded function with a Lipschitz constant of 1. For any $\delta>0$, we have:

$$\|\widehat{\mathscr{K}}-\mathscr{K}\|_\infty\leq\delta,$$

where $\widehat{\mathscr{K}}$ is a random-feature approximation of $\mathscr{K}$, with any constant probability as long as $M\geq\mathscr{C}\frac{d}{\delta^2}\log(\frac{\mathrm{diam}(\mathscr{M})\mathbb{E}\boldsymbol{\omega}^\top\boldsymbol{\omega}}{\delta})$ where $\mathscr{C}$ is an absolute constant and $\mathscr{M}$ is the smallest ball containing all vectors of the form $\mathbf{z}=\frac{\mathbf{Q}_i}{d^{\frac{1}{4}}}-\frac{\mathbf{K}_j}{d^{\frac{1}{4}}}$. Since $\|\mathbf{Q}_i\|,\|\mathbf{K}_j\|\leq R$, we conclude that $\|\mathbf{z}\|\leq\frac{2R}{d^{\frac{1}{4}}}$ and thus one can take $\mathrm{diam}(\mathscr{M})=\frac{4R}{d^{\frac{1}{4}}}$. Further, $\mathbb{E}\boldsymbol{\omega}^\top\boldsymbol{\omega}=d$ since $\boldsymbol{\omega}$ is marginally a standard multivariate Gaussian vector. For all $1\leq i,j\leq L$, we have:

$$\begin{aligned}|\widehat{\mathbf{A}}_{i,j}-\mathbf{A}_{i,j}|&=\|(\mathbf{D}_\mathbf{Q})_{i,i}(\widehat{\mathscr{K}_{i,j}}-\mathscr{K}_{i,j})(\mathbf{D}_\mathbf{K})_{j,j}\|_\infty\leq(\mathbf{D}_\mathbf{Q})_{i,i}\|\widehat{\mathscr{K}}-\mathscr{K}\|_\infty(\mathbf{D}_\mathbf{K})_{j,j}\\&\leq\delta\exp(d^{-1/2}R^2)\end{aligned}$$

since $(\mathbf{D}_\mathbf{K})_{i,i},(\mathbf{D}_\mathbf{Q})_{j,j}\leq\exp(d^{-1/2}R^2/2)$. Taking $\delta=\exp(-d^{-1/2}R^2)\varepsilon$ completes the proof. □

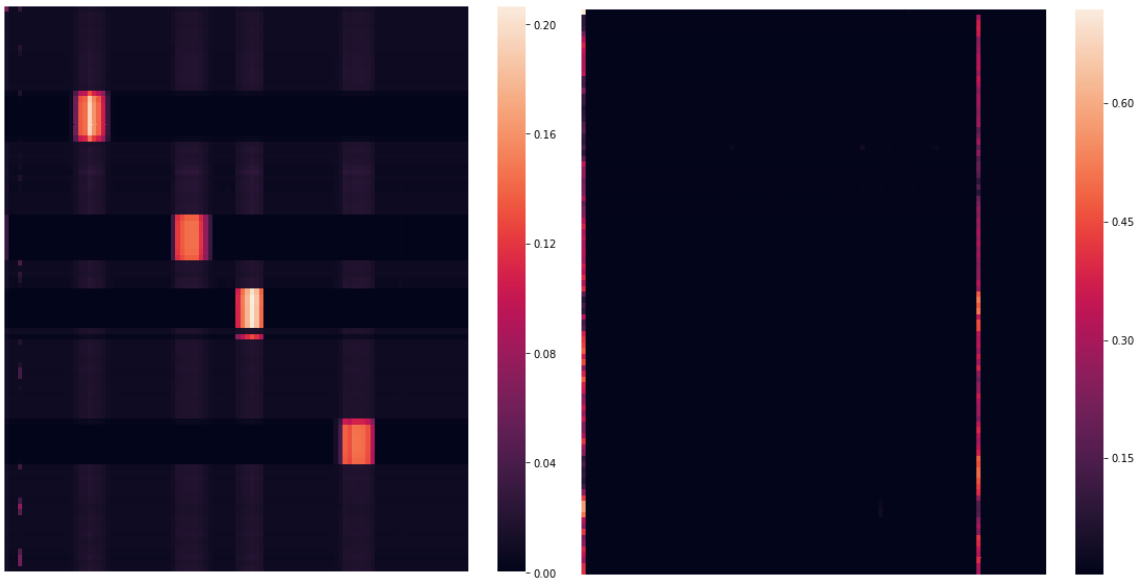


Fig. 3.14 Two self-attention heads in more detail. **(1)** Head 1-2 (second layer, third head), **(2)** Head 4-1 (fifth layer, second head). Note the block attention in Head 1-2 and the vertical attention (to the start token ('M') and the 85th token ('C')) in Head 4-1.

Appendix 3.B Hyperparameters

All Performer and Transformer runs used 0.5 gradient clipping (Zhang et al., 2020a), 0.1 weight decay (Krogh and Hertz, 1991), 0.1 dropout (Srivastava et al., 2014), 10^{-3} learning rate with Adam (Kingma and Ba, 2015) hyperparameters $\beta_1 = 0.9, \beta_2 = 0.98, \varepsilon = 10^{-9}$, with batch size maximized until TPU memory overload for a specific model. For the Reformer (Kitaev et al., 2020), we used the same hyperparameters as mentioned for protein experiments without gradient clipping, and hyperparameters from (Kitaev et al., 2020) for the ImageNet64 experiment. In both cases, the Reformer used the same default LSH self-attention parameters.

All 36-layer protein experiments used the same amount of compute (256 TPUs, 8 gigabytes per chip). For concatenated experiments, 256 TPUs were also used for the Performer, while 64 TPUs were used for the 1-3 layer Transformer model (using 256 TPUs did not make a difference in accuracy).

For all Performer training experiments with FAVOR or GA, we use $M = 256$, orthogonal random features and, in the case of GA, map $=$ ReLU is the default kernel choice.

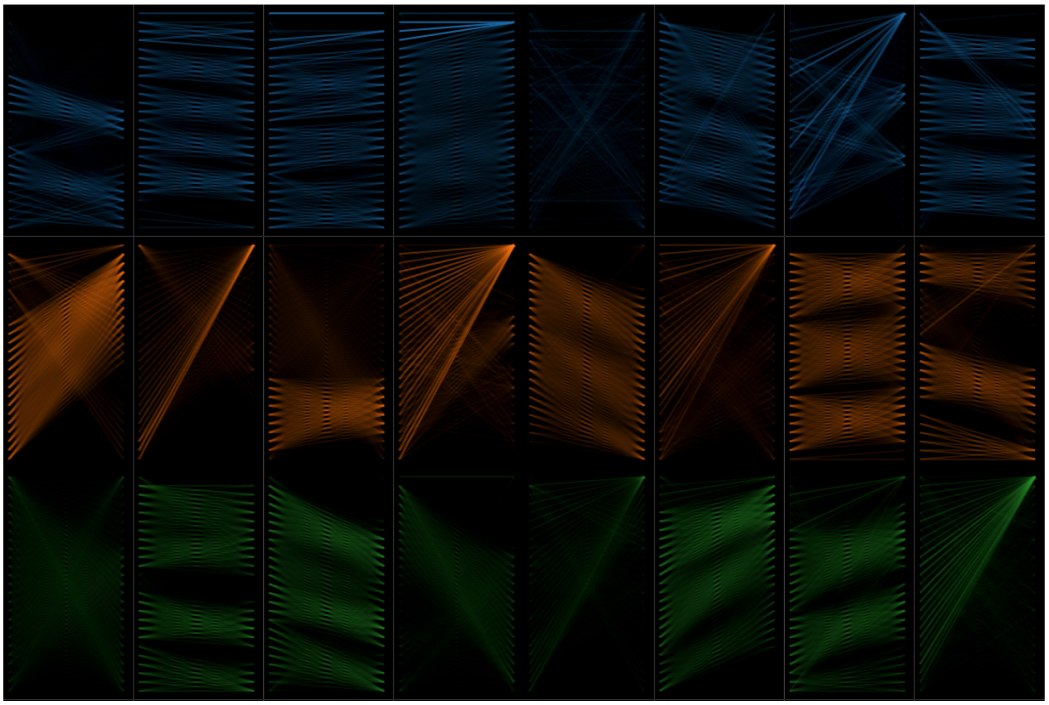


Fig. 3.15 The self-attention patterns on the first 25 tokens. The illustration is based on (Vig, 2019; Vig and Belinkov, 2019). Note that, similar to prior work on protein Transformers (Madani et al., 2020), the attention matrices include both local and global patterns.

Appendix 3.C Experimental details for protein modelling

3.C.1 TrEMBL dataset

We used the TrEMBL part of proteins dataset[1] which contains 139,394,261 sequences of which 106,030,080 are unique. While the training dataset appears smaller than the one used in (Madani et al., 2020), we argue that it includes most of the relevant sequences. Specifically, the TrEMBL dataset consists of the subset of UniProtKB sequences that have been computationally analyzed but not manually curated, and accounts for $\approx 99.5\%$ of the total number of sequences in the UniProtKB dataset[2].

Following the methodology described in (Madani et al., 2020), we used both an OOD-Test set, where a selected subset of Pfam families are held-out for valuation, and an IID split, where the remaining protein sequences are split randomly into train, valid, and test tests. We held-out the following protein families (PF18369, PF04680, PF17988, PF12325, PF03272, PF03938, PF17724, PF10696, PF11968, PF04153, PF06173, PF12378, PF04420, PF10841,

[1]https://www.uniprot.org/statistics/TrEMBL
[2]https://www.uniprot.org/uniprot/

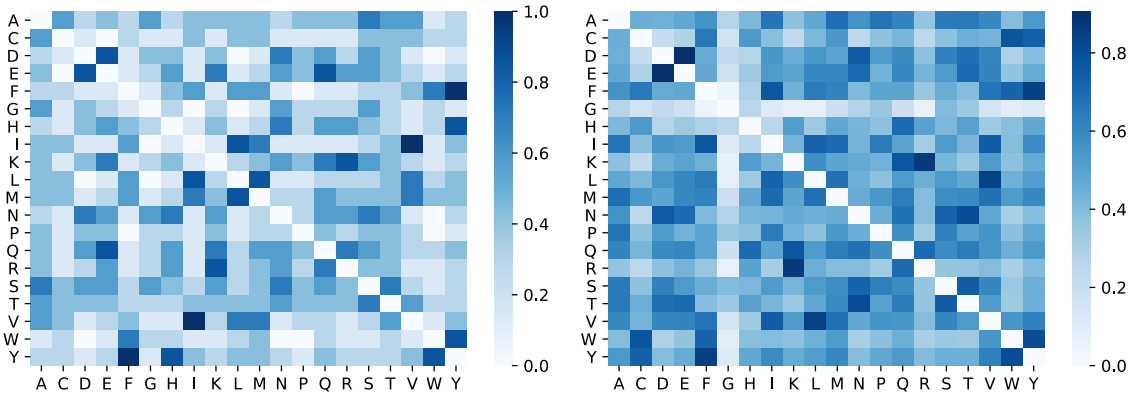


Fig. 3.16 Amino acid similarity matrix estimated from self-attention matrices aggregated across a small subset of sequences, as described in (Vig et al., 2020). **(1)** the normalized BLOSUM matrix, **(2)** the amino acid similarity estimated via the trained Performer model.

Dataset	**Set Name**	**Count**	**Length Statistics**				
			Min	**Max**	**Mean**	**STD**	**Median**
TrEMBL	Train	104,863,744	2	74,488	353.09	311.16	289.00
	Valid	102,400	7	11,274	353.62	307.42	289.00
	Test	1,033,216	8	32,278	353.96	312.23	289.00
	OOD	29,696	24	4,208	330.96	269.86	200.00
TrEMBL (concat)	Train	4,532,224	8,192	8,192	8,192	0	8,192
	Valid	4,096					

Table 3.2 Statistics for the TrEMBL single sequence and the long sequence task.

PF06917, PF03492, PF06905, PF15340, PF17055, PF05318), which resulted in 29,696 OOD sequences. We note that, due to deduplication and potential TrEMBL version mismatch, our OOD-Test set does not match exactly the one in (Madani et al., 2020). We also note that this OOD-Test selection methodology does not guarantee that the evaluation sequences are within a minimum distance from the sequences used during training. In future work, we will include rigorous distance based splits.

The statistics for the resulting dataset splits are reported in Table 3.2. In the standard sequence modeling task, given the length statistics that are reported in the table, we clip single sequences to maximum length $L = 1024$, which results in few sequences being truncated significantly.

In the long sequence task, the training and validation sets are obtained by concatenating the sequences, separated by an end-of-sequence token, and grouping the resulting chain into non-overlapping sequences of length $L = 8192$.

3.C.2 Empirical baseline

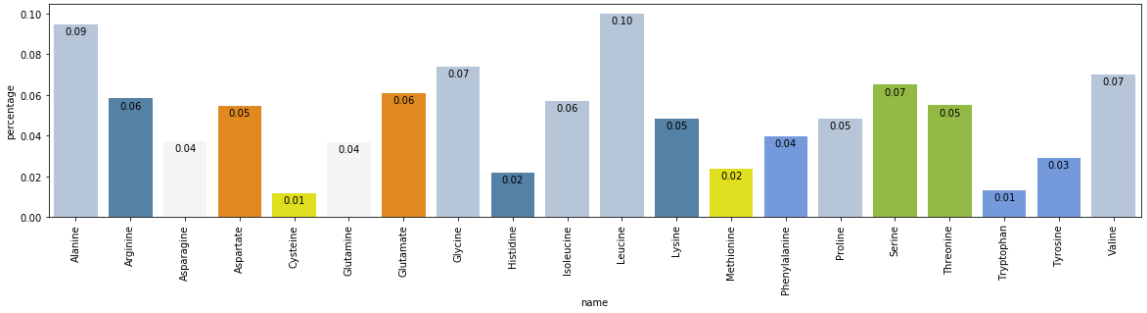


Fig. 3.17 Visualization of the estimated empirical distribution for the 20 standard amino acids, colored by their class.

A random baseline, with uniform probability across all the vocabulary tokens at every position, has accuracy 5% (when including only the 20 standard amino acids) and 4% (when also including the 5 anomalous amino acids (Consortium, 2019)). However, the empirical frequencies of the various amino acids in our dataset may be far from uniform, so we also consider an *empirical baseline* where the amino acid probabilities are proportional to their empirical frequencies in the training set.

Figure 3.17 shows the estimated empirical distribution. We use both the standard and anomalous amino acids, and we crop sequences to length 1024 to match the data processing performed for the Transformer models. The figure shows only the 20 standard amino acids, colored by their class, for comparison with the visualization on the TrEMBL web page[1].

[1] https://www.uniprot.org/statistics/TrEMBL

Chapter 4

FAVOR+: positive random features

4.1 Motivation

Let's take a close look at Algorithm 3. In practice, an important characteristic of any algorithm is not only its computational efficiency but also its arithmetic stability. Typically, division operations are sources of arithmetic instabilities, and there are divisions in the algorithm's last line where $\mathrm{diag}(\mathbf{Buf}^{(3)}_{:,d+1})^{-1}\mathbf{Buf}^{(3)}_{:,:d}$ is computed. In the implementation, to compute this matrix product we simply divide each column of $\mathbf{Buf}^{(3)}_{:,:d} \in \mathbb{R}^{L\times d}$ by $\mathbf{Buf}^{(3)}_{:,d+1} \in \mathbb{R}^{L}$.

Consider the case onCausal = False, onGA = True first. By the definition of $\mathbf{Buf}^{(2)}$, we have:

$$\mathbf{Buf}^{(2)}_{:,d+1} = \mathbf{P}^{(1)}(\mathbf{P}^{(2)})^{\top}\mathbf{1}_L \tag{4.1}$$

where $\mathbf{P}^{(1)}, \mathbf{P}^{(2)}$ are row-wise applications of $g^{(1)}, g^{(2)}$ respectively. If $g^{(1)}, g^{(2)}$ can potentially have negative entries, $\mathbf{P}^{(1)}, \mathbf{P}^{(2)}$ can also get negative entries. Then, according to (4.1), $\mathbf{Buf}^{(2)}_{:,d+1}$ can potentially take negative entries. This contradicts to the nature of $\mathbf{Buf}^{(2)}_{:,d+1}$ which is meant to be a renormalization of $\mathbf{Buf}^{(2)}_{:,:d}$ so that the result of Algorithm 3 is bounded. Moreover, since the signs of $\mathbf{P}^{(1)}, \mathbf{P}^{(2)}$ can be different, each entry of $\mathbf{Buf}^{(2)}_{:,d+1}$ can be a sum of positive and negative numbers, and the result can become very small in magnitude. Division by small values is unstable in practice, since it can lead to overflows in the result. For this reason, in Figure 3.11, cos and tanh versions of Performer halt with NaNs very early during training. Moreover, the negative denominator is inconsistent with the conceptual idea of self-attention as assigning positive weights which sum to 1.

In the case of GA, this problem can be fixed by restricting $g^{(1)}, g^{(2)}$ to always map into vectors with positive values: $g^{(1)}, g^{(2)} : \mathbb{R}^d \to \mathbb{R}^M_+$. Then, the renormalizer $\mathbf{Buf}^{(2)}_{:,d+1}$ is always positive-valued. Further, there is no effect of collapsing positive and negative terms, hence, the magnitude of each entry of $\mathbf{Buf}^{(2)}_{:,d+1}$ cannot become too small. Note that for Performer-

ReLU, values of $g^{(1)}, g^{(2)}$ are already non-negative. We can further add a small $\varepsilon > 0$ to the entries of $g^{(1)}, g^{(2)}$ so that they are strictly positive. This solution of positive-valued $g^{(1)}, g^{(2)}$ also extends to onCausal = True.

In the case of FAVOR (onGA = False in Algorithm 3), we can get the same problem, since $\mathbf{P}^{(1)}, \mathbf{P}^{(2)}$ are complex numbers with potentially arbitrary magnitudes and arguments. Hence, $\mathbf{Buf}^{(2)}_{:,d+1}$, before taking its real part, can potentially have arbitrary complex entries with negative and/or very small real parts which can result in instabilities during training. Similarly to GA, having $f^{(1)}, f^{(2)}$ mapping into positive values would solve the problem. Unfortunately, by modifying $f^{(1)}, f^{(2)}$ we can lose the unbiased Gaussian kernel approximation.

In this chapter, we propose solution: a mathematical insight which allows a construction of positive-valued random features for the Gaussian kernel. We find that these random features are also compatible with block-orthogonal random features, which results in the FAVOR+ mechanism: *F*ast *A*ttention *V*ia *positive O*rthogonal *R*andom features. The chapter is structured as follows:

- In Section 4.2, we define positive random features and FAVOR+.
- In Section 4.3, we provide extensive theoretical analysis of FAVOR+.
- In Section 4.4, we provide intuition for the theoretical results from Section 4.3. We provide generalizations formulated through *beautiful functions* – functions which can be represented as expectations of power series with nonnegative coefficients. This appears to be a key property for obtaining concentration improvements when using block-orthogonal random features, and the Gaussian kernel is a special case of such functions.
- In Section 4.5, we describe a *masking mechanism* which allows injection of input data priors into the FAVOR+ and GA, while maintaining efficient evaluation of self-attention.
- In Section 4.6, we present extensive empirical evaluations of FAVOR+ which include large-scale training setups.
- Section 4.7 is reserved for discussions.

4.2 FAVOR+: positive random features for the Gaussian kernel

Consider $f^{(1)}_{\text{pos}}, f^{(2)}_{\text{pos}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined as follows:

$$f^{(1)}_{\text{pos}}(\boldsymbol{\omega}, \mathbf{x}) = \exp\left(\boldsymbol{\omega}^\top \mathbf{x} - \|\mathbf{x}\|^2\right), \quad f^{(2)}_{\text{pos}}(\boldsymbol{\omega}, \mathbf{y}) = \exp\left(\boldsymbol{\omega}^\top \mathbf{y} - \|\mathbf{y}\|^2\right). \tag{4.2}$$

The theorem below proves that these functions, equipped with the standard multivariate Gaussian distribution $p_{\text{SG}}(\boldsymbol{\omega})$, are proper random features for the Gaussian kernel (proof is in the Appendix):

Theorem 5. *$(p_{\text{SG}}, f^{(1)}_{\text{pos}}, f^{(2)}_{\text{pos}})$ are random features for the Gaussian kernel, i.e. for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,*

$$K(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{p_{\text{SG}}(\boldsymbol{\omega})} \operatorname{Re}\left(f^{(1)}_{\text{pos}}(\boldsymbol{\omega}, \mathbf{x}) f^{(2)}_{\text{pos}}(\boldsymbol{\omega}, \mathbf{y})\right) = \mathbb{E}_{p_{\text{SG}}(\boldsymbol{\omega})}\left(f^{(1)}_{\text{pos}}(\boldsymbol{\omega}, \mathbf{x}) f^{(2)}_{\text{pos}}(\boldsymbol{\omega}, \mathbf{y})\right) \tag{4.3}$$

where we can get rid of the $\operatorname{Re}(\cdot)$ since both $f^{(1)}_{\text{pos}}, f^{(2)}_{\text{pos}}$ map into real numbers. The variance of these random features has the following form:

$$\operatorname{Var}_{p_{\text{SG}}(\boldsymbol{\omega})}\left(f^{(1)}_{\text{pos}}(\boldsymbol{\omega}, \mathbf{x}) f^{(2)}_{\text{pos}}(\boldsymbol{\omega}, \mathbf{y})\right) = \exp(4\mathbf{x}^\top \mathbf{y}) - K(\mathbf{x}, \mathbf{y})^2. \tag{4.4}$$

The most important property of $f^{(1)}_{\text{pos}}, f^{(2)}_{\text{pos}}$ is that they map into positive numbers $\mathbb{R}_+$ since $\exp(\cdot)$ of a real number is always positive (4.3). For this reason, we refer to $(p_{\text{SG}}, f^{(1)}_{\text{pos}}, f^{(2)}_{\text{pos}})$ as *positive random features* (PosRFs). Figure 4.1 compares variance of PosRFs and TrigRFs (2.10). We observe that variance of PosRFs converges to zero for any length when the angle between $\mathbf{x}$ and $\mathbf{y}$ approaches π. The same happens to TrigRFs when the angle approaches 0. Hence, different types of random features can be practical in different scenarios. For larger norms of $\mathbf{x}$ and $\mathbf{y}$, when the angle between them is close to 0, the variance of PosRFs grows exponentially. This can be thought as a price paid for positive-valued features.

We can use $f^{(1)}_{\text{pos}}, f^{(2)}_{\text{pos}}$ for the Monte-Carlo approximation of Gaussian kernel as described in Section 2.1.2. Further, instead of using i.i.d. $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(M)}$, we can again generate block-orthogonal $\boldsymbol{\omega}^{(m)}$'s using Algorithm 2. Since $\boldsymbol{\omega}^{(m)}$'s are marginally coming from the standard Gaussian distribution, the approximation is still unbiased, as mentioned in Section 2.1.4. Furthermore, in the next section, we show that this strategy leads to provable variance reduction. Plugging $f^{(1)}_{\text{pos}}, f^{(2)}_{\text{pos}}$ into Algorithm 3 results in the next iteration of the efficient self-attention mechanisms, *F*ast *A*ttention *V*ia *positive O*rthogonal *R*andom features (FAVOR+).

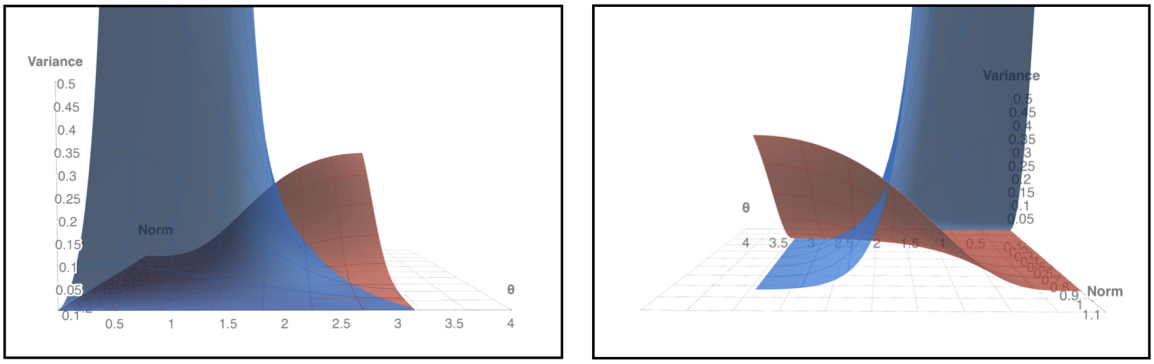


Fig. 4.1 Variance of PosRFs (blue) and TrigRFs (red) expressed as the function of Norm of $\mathbf{x}$ and $\mathbf{y}$ and angle θ between $\mathbf{x}$ and $\mathbf{y}$. Both variances can be expressed through Norm and θ since $\mathbf{x}^\top \mathbf{y} = \mathrm{Norm}^2 \cos(\theta)$, $K(\mathbf{x},\mathbf{y}) = \exp(-\mathrm{Norm}^2(1-\cos(\theta)))$.

The next section is dedicated to theoretical analysis of FAVOR+.

4.3 Concentration analysis

As with FAVOR (Section 3.5), we can provide a theoretical analysis of the variance and concentration for FAVOR+. We first start with the analog of Theorem 3 for PosRFs:

Theorem 6. *Let $\widehat{\mathbf{x}},\widehat{\mathbf{y}}$ be defined according to (2.5) where $f^{(1)} = f^{(1)}_{\mathrm{pos}}, f^{(2)} = f^{(2)}_{\mathrm{pos}}$ and $\boldsymbol{\omega}^{(1)},\ldots,\boldsymbol{\omega}^{(M)}$ are i.i.d. samples from $\mathcal{N}(0,1)^d$. Further, let $\widehat{\mathbf{x}}_{\mathrm{ort}},\widehat{\mathbf{y}}_{\mathrm{ort}}$ be defined according to (2.11) where $f^{(1)} = f^{(1)}_{\mathrm{pos}}, f^{(2)} = f^{(2)}_{\mathrm{pos}}$ and $\boldsymbol{\Omega}$ is generated by Algorithm 2. If $M \leq d$, then*

$$\mathrm{Var}\,\widehat{\mathbf{x}}^\top_{\mathrm{ort}}\widehat{\mathbf{y}}_{\mathrm{ort}} \leq \mathrm{Var}\,\widehat{\mathbf{x}}^\top\widehat{\mathbf{y}} - \frac{2(M-1)}{M(d+2)}\left(K(\mathbf{x},\mathbf{y}) - \exp\left(-\|\mathbf{x}\|^2 - \|\mathbf{y}\|^2\right)\right)^2.$$

If $M > d$ and M/d is integer, then

$$\mathrm{Var}\,\widehat{\mathbf{x}}^\top_{\mathrm{ort}}\widehat{\mathbf{y}}_{\mathrm{ort}} \leq \mathrm{Var}\,\widehat{\mathbf{x}}^\top\widehat{\mathbf{y}} - \frac{2(d-1)}{M(d+2)}\left(K(\mathbf{x},\mathbf{y}) - \exp\left(-\|\mathbf{x}\|^2 - \|\mathbf{y}\|^2\right)\right)^2.$$

Hence, we conclude that the orthogonal variant of PosRFs results in a strictly smaller variance than i.i.d. variant. As we mentioned in Section 2.1.4, using orthogonal variants comes with negligible additional computations, hence the choice for orthogonal RFs in FAVOR+ is justified. Importantly, this result is universal, i.e. it holds for any dimensionality d, not just asymptotically for d large enough as it was the case for TrigRFs (Theorem 3). This bound is attainable in the case of PosRFs since all terms of Taylor expansion of $\exp(\cdot)$

are positive. This is not the case for TrigRFs with $\mathrm{Re}\,(\exp(\mathrm{i}\cdot)) = \cos(\cdot)$. More details are provided in the next section.

PosRFs are unbounded in general, meaning that, when $\boldsymbol{\omega}$ is very big and the dot products $\boldsymbol{\omega}^\top \mathbf{x}$, $\boldsymbol{\omega}^\top \mathbf{y}$ are positive, $f^{(1)}_{\mathrm{pos}}, f^{(2)}_{\mathrm{pos}}$ from (4.3) can potentially take very big values. Because of this, we were not able to provide exponentially tight concentration results similar to Theorem 4. However, we can modify the definition of PosRFs (4.3) in such a way that the random features approximate Gaussian kernel with a small quantifiable bias, but we then are able to provide one-sided concentration results.

Define $f^{(1)}_{\mathrm{reg}}, f^{(2)}_{\mathrm{reg}}$ as follows:

$$f^{(1)}_{\mathrm{reg}}(\boldsymbol{\omega}, \mathbf{x}) = \exp\left(\frac{\sqrt{d}}{\|\boldsymbol{\omega}\|}\boldsymbol{\omega}^\top \mathbf{x} - \|\mathbf{x}\|^2\right), \quad f^{(2)}_{\mathrm{reg}}(\boldsymbol{\omega}, \mathbf{y}) = \exp\left(\frac{\sqrt{d}}{\|\boldsymbol{\omega}\|}\boldsymbol{\omega}^\top \mathbf{y} - \|\mathbf{y}\|^2\right). \quad (4.5)$$

We refer to the triple $(p_{\mathrm{SG}}, f^{(1)}_{\mathrm{reg}}, f^{(2)}_{\mathrm{reg}})$ as *regularized random features* (RegRFs) since $\boldsymbol{\omega}$ is "reguralized" by normalizing its length. Note that RegRFs are bounded since $\boldsymbol{\omega}^\top \mathbf{x}/\|\boldsymbol{\omega}\| \leq \|\mathbf{x}\|$. These random features do not approximate the Gaussian kernel though. Instead, they approximate the following kernel defined for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ through $f^{(1)}_{\mathrm{reg}}, f^{(2)}_{\mathrm{reg}}$:

$$K^{\mathrm{reg}}(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})} f^{(1)}_{\mathrm{reg}}(\boldsymbol{\omega}, \mathbf{x}) f^{(2)}_{\mathrm{reg}}(\boldsymbol{\omega}, \mathbf{y}). \quad (4.6)$$

Definition (4.5) is very similar to PosRFs (4.3) with the difference that a scaling factor $\sqrt{d}/\|\boldsymbol{\omega}\|$ is added under the exponent. Note that, if $\boldsymbol{\omega} \sim \mathcal{N}(0,1)^d$, $\mathbb{E}\|\boldsymbol{\omega}\|^2 = d$. Hence, presumably, $f^{(1)}_{\mathrm{reg}}, f^{(2)}_{\mathrm{reg}}$ shouldn't be "too different" from $f^{(1)}_{\mathrm{pos}}, f^{(2)}_{\mathrm{pos}}$. Analogously, $K^{\mathrm{reg}}(\mathbf{x}, \mathbf{y})$ shouldn't differ "too much " from $K(\mathbf{x}, \mathbf{y})$. This intuitive observation is formalized in the following theorem:

Theorem 7. *Consider the set of pairs $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, $d \in \mathbb{N}$, such that $\max(\|\mathbf{x}\|^2, \|\mathbf{y}\|^2, \mathbf{x}^\top \mathbf{y}) \leq \mathscr{C}$ where $\mathscr{C} \geq 0$ is some constant. Then, as d grows, the following holds for all $\mathbf{x}, \mathbf{y}$:*

$$\frac{K^{\mathrm{reg}}(\mathbf{x}, \mathbf{y})}{K(\mathbf{x}, \mathbf{y})} \geq 1 - \frac{2}{d^{\frac{1}{3}}} + o\left(\frac{1}{d^{\frac{1}{3}}}\right), \quad \frac{K^{\mathrm{reg}}(\mathbf{x}, \mathbf{y})}{K(\mathbf{x}, \mathbf{y})} \leq 1.$$

Furthermore, the latter holds for $d \geq 2$ even if the condition $\max(\|\mathbf{x}\|^2, \|\mathbf{y}\|^2, \mathbf{x}^\top \mathbf{y}) \leq \mathscr{C}$ is not satisfied, i.e. $K^{\mathrm{reg}}(\mathbf{x}, \mathbf{y})$ is a universal lower bound for the Gaussian kernel $K(\mathbf{x}, \mathbf{y})$.

This theorem states that, essentially, $K^{\mathrm{reg}}(\mathbf{x}, \mathbf{y})$ converges to the Gaussian kernel $K(\mathbf{x}, \mathbf{y})$ when d is big enough.

For $\mathbf{x},\mathbf{y}\in\mathbb{R}^d$, define $\widehat{\mathbf{x}}_{\mathrm{reg}},\widehat{\mathbf{y}}_{\mathrm{reg}}\in\mathbb{R}^M$ as random vectors of the form:

$$\widehat{\mathbf{x}}_{\mathrm{reg}}=M^{-1/2}(f_{\mathrm{reg}}^{(1)}(\boldsymbol{\omega}^{(m)},\mathbf{x}))_{m=1}^{M},\quad \widehat{\mathbf{y}}_{\mathrm{reg}}=M^{-1/2}(f_{\mathrm{reg}}^{(2)}(\boldsymbol{\omega}^{(m)},\mathbf{y}))_{m=1}^{M}$$

where $\boldsymbol{\omega}^{(1)},\ldots,\boldsymbol{\omega}^{(M)}$ are i.i.d. samples from $\mathcal{N}(0,1)^d$. Let $\widehat{\mathbf{x}}_{\mathrm{ortreg}},\widehat{\mathbf{y}}_{\mathrm{ortreg}}$ be defined in the same way as $\widehat{\mathbf{x}}_{\mathrm{reg}},\widehat{\mathbf{y}}_{\mathrm{reg}}$ but with $\boldsymbol{\omega}^{(1)},\ldots,\boldsymbol{\omega}^{(M)}$ sampled by Algorithm 2. Then, both $(\widehat{\mathbf{x}}_{\mathrm{reg}})^\top\widehat{\mathbf{y}}_{\mathrm{reg}}$ and $(\widehat{\mathbf{x}}_{\mathrm{ortreg}})^\top\widehat{\mathbf{y}}_{\mathrm{ortreg}}$ are unbiased random feature approximations of $K^{\mathrm{reg}}(\mathbf{x},\mathbf{y})$. Denote by $\mathcal{M}_Z(\phi)=\mathbb{E}(\exp(\phi Z))$ a moment generating function of the random variable Z. The next result provides exponentially small bounds for upper tails of this approximation, demonstrating that RegRFs can approximate small kernel values especially well:

Theorem 8. *Let $\mathbf{x},\mathbf{y}\in\mathbb{R}^d$. The following holds for any $\phi>0$, $a>K^{\mathrm{reg}}(\mathbf{x},\mathbf{y})$ and $M\leq d$:*

$$\mathbb{P}\left((\widehat{\mathbf{x}}_{\mathrm{reg}})^\top\widehat{\mathbf{y}}_{\mathrm{reg}}>a\right)\leq\exp(-\phi Ma)\mathcal{M}_Z(\phi)^M,$$

$$\mathbb{P}\left((\widehat{\mathbf{x}}_{\mathrm{ortreg}})^\top\widehat{\mathbf{y}}_{\mathrm{ortreg}}>a\right)\leq\exp(-\phi Ma)\Bigg(\mathcal{M}_Z(\phi)^M - \exp\left(-M(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2)\right)\frac{\phi^4M(M-1)}{4(d+2)}\|\mathbf{x}+\mathbf{y}\|^4\Bigg)$$

where $Z=f_{\mathrm{reg}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{reg}}^{(2)}(\boldsymbol{\omega},\mathbf{y})$, $\boldsymbol{\omega}\sim\mathcal{N}(0,1)^d$.

While these results hold for RegRFs and the kernel $K^{\mathrm{reg}}(\mathbf{x},\mathbf{y})$ which converges to $K(\mathbf{x},\mathbf{y})$ when $d\to+\infty$ (Theorem 7), we cannot prove the same or similar result for PosRFs since the moment generating function $\mathcal{M}_Z(\phi)$ doesn't exist. This is because PosRFs are unbounded. This problem is addressed in Chapter 6 where we propose a new type of positive random features for the Gaussian kernel which has a smaller variance and enjoys exponentially fast concentration similar to FAVOR (Theorem 4). We note that there exist techniques for bounding the concentration of unbounded RFs (Chamakh et al., 2020). While these techniques are applicable to functions with polynomial growth, perhaps there is a way to extend them for exponentially growing functions such as $f_{\mathrm{pos}}^{(\cdot)}$. We leave these extensions to future work.

The next section provides an intuition about proofs of Theorems 6 and 8.

4.4 Beautiful functions and generalizations of Theorems 6, 8

In this section we provide much more general theoretical results which imply and provide intuition for Theorem 6 and Theorem 8. We need the following definition:

Definition 2. *We say that function $\mathscr{F} : \mathbb{R}^d \to \mathbb{R}$ is beautiful if $\mathscr{F}$ can be expressed as:*

$$\mathscr{F}(\mathbf{z}) = \mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z}) = \mathbb{E}_{\boldsymbol{\omega}\sim\Omega}[\mathscr{G}(\boldsymbol{\omega}^\top \mathbf{z})], \tag{4.7}$$

where Ω is an isotropic (i.e. invariant with respect to rotations around $\mathbf{0}_d$) distribution on $\mathbb{R}^d$ and $\mathscr{G} : \mathbb{R} \to \mathbb{R}$ can be expressed as $\mathscr{G}(z) = \sum_{k=0}^{\infty} a_k z^k$ for every $z \in \mathbb{R}$ with $a_k \geq 0$ for $k = 0, 1, \ldots$. In the formula above, we assume that the expectation on the right hand side exists.

It appears that beautiful functions can be used to define $K(\cdot,\cdot)$ and $K^{\mathrm{reg}}(\cdot,\cdot)$:

Remark 1. *If one takes $\Omega = \mathscr{N}(0,1)^d$ (note that this is an isotropic distribution) and $\mathscr{G}(z) = \exp(z - \|\mathbf{x}\|^2 - \|\mathbf{y}\|^2) = \sum_{k=0}^{\infty} \exp(-\|\mathbf{x}\| - \|\mathbf{y}\|^2)\frac{z^k}{k!}$ satisfying conditions from Definition 2, then, according to Theorem 5, the following is true for $\mathbf{z} = \mathbf{x} + \mathbf{y}$:*

$$K(\mathbf{x},\mathbf{y}) = \mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z}). \tag{4.8}$$

Similarly, $K^{\mathrm{reg}}(\mathbf{x},\mathbf{y}) = \mathscr{F}_{\Omega_{\mathrm{reg}},\mathscr{G}}(\mathbf{z})$, where Ω_{reg} stands for the uniform distribution on the sphere of radius $\sqrt{d}$ with center at $\mathbf{0}_d$. Ω_{reg} is clearly isotropic and is the result of reparametrization $\boldsymbol{\omega} \to \frac{\sqrt{d}}{\|\boldsymbol{\omega}\|}\boldsymbol{\omega}$, $\boldsymbol{\omega} \sim \mathscr{N}(0,1)^d$.

The condition of nonnegative coefficients in the power series will be crucial in the proof. It holds for PosRFs as mentioned in the remark above. It, however, doesn't hold for TrigRFs since $\mathscr{G}(z) = \mathrm{Re}\,(\exp(\mathrm{i}z)) = \cos(z)$ in this case. This is the reason why it's only possible to provide asymptotic improvements when using block-orthogonal TrigRFs for big values of d (Theorem 3).

We consider two estimators of the beautiful functions from Definition 2 that directly lead (through Remark 1) to PosRFs and RegRFs with i.i.d. and block-orthogonal variants. Standard Monte Carlo estimator samples independently $\boldsymbol{\omega}^{(\mathrm{iid},1)}, \ldots, \boldsymbol{\omega}^{(\mathrm{iid},M)} \sim \Omega$ and then computes:

$$\widehat{\mathscr{F}}^{\mathrm{iid}}_M(\mathbf{z}) = \frac{1}{M}\sum_{m=1}^{M} \mathscr{G}((\boldsymbol{\omega}^{(\mathrm{iid},m)})^\top \mathbf{z}). \tag{4.9}$$

Orthogonal Monte Carlo estimator samples $\boldsymbol{\omega}^{(\mathrm{ort},1)}, \ldots, \boldsymbol{\omega}^{(\mathrm{ort},M)}$ in such a way that marginally we have: $\boldsymbol{\omega}^{(\mathrm{ort},m)} \sim \Omega$, but $(\boldsymbol{\omega}^{(\mathrm{ort},m_1)})^\top \boldsymbol{\omega}^{(\mathrm{ort},m_2)} = 0$ for all $m_1 \neq m_2$, $\lfloor m_1/d \rfloor = \lfloor m_2/d \rfloor$. Also, blocks of m consecutive $\boldsymbol{\omega}^{(ort,m)}$'s are mutually independent. Such a block-orthogonal ensemble can be always created if Ω is isotropic, e.g. by Algorithm 2 for $\Omega = \mathscr{N}(0,1)^d$ or

by reparametrizing $\boldsymbol{\omega}$'s ($\boldsymbol{\omega} \rightarrow \frac{\sqrt{d}}{\|\boldsymbol{\omega}\|}\boldsymbol{\omega}$) from Algorithm 2 for Ω_{reg}. We define:

$$\widehat{\mathscr{F}}^{\mathrm{ort}}_{M}(\mathbf{z}) = \frac{1}{M}\sum_{m=1}^{M}\mathscr{G}((\boldsymbol{\omega}^{(\mathrm{ort},m)})^{\top}\mathbf{z}). \tag{4.10}$$

The estimators of beautiful functions based on standard Monte Carlo procedure using independent vectors $\boldsymbol{\omega}^{(\mathrm{iid},1)},\ldots,\boldsymbol{\omega}^{(\mathrm{iid},M)}$ guarantee strong concentration bounds since independent $\boldsymbol{\omega}^{(\mathrm{iid},m)}$'s provide a way to obtain exponentially small upper bounds on failure probabilities through moment generating functions. This classic observation can be summarized in the following result:

Theorem 9. *Consider an estimator $\widehat{\mathscr{F}}^{\mathrm{iid}}_{M}(\mathbf{z})$ of the beautiful function $\mathscr{F}_{\Omega,\mathscr{G}}$ evaluated at $\mathbf{z}$. Then the following holds for any $\phi > 0$, $a > \mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z})$:*

$$\mathbb{P}(\widehat{\mathscr{F}}^{\mathrm{iid}}_{M}(\mathbf{z}) > a) \leq \exp(-\phi M a)\mathscr{M}_{Z}(\phi)^{M},$$

where $Z = \mathscr{G}(\boldsymbol{\omega}^{\top}\mathbf{z})$, $\boldsymbol{\omega} \sim \Omega$.

The above result gives exponentially small upper bounds on tail probabilities for the i.i.d. estimator. Below we provide two main theoretical results for the block-orthogonal estimator.

Theorem 10. *If $\mathscr{F}_{\Omega,\mathscr{G}}$ is a beautiful function then the following holds for $M \leq d$, Z as in Theorem 9 and any $\phi > 0$, $a > \mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z})$:*

$$\mathbb{P}(\widehat{\mathscr{F}}^{\mathrm{ort}}_{M}(\mathbf{z})) > a) \leq \exp(-\phi M a)\left(\mathscr{M}_{Z}(\phi)^{M} - \frac{\phi^{4}M(M-1)}{4d^{2}(d+2)}a_{1}^{2}a_{0}^{M-2}\|\mathbf{z}\|^{4}\left(\mathbb{E}\|\boldsymbol{\omega}\|^{2}\right)^{2}\right)$$

This result shows that features obtained from the ensembles of block-orthogonal random vectors provide exponentially small bounds on tail probabilities and that these bounds are better than for estimators using i.i.d. random features.

We also obtain a similar result for variances of the considered estimators:

Theorem 11. *If $\mathscr{F}_{\Omega,\mathscr{G}}$ is a beautiful function, then the following holds. If $M \leq d$, then*

$$\mathrm{Var}\,\widehat{\mathscr{F}}^{\mathrm{ort}}_{M}(\mathbf{z}) \leq \mathrm{Var}\,\widehat{\mathscr{F}}^{\mathrm{iid}}_{M}(\mathbf{z}) - \frac{2(M-1)}{M(d+2)}(\mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z}) - \mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{0}_{d}))^{2}.$$

If $M > d$ and M/d is integer, then

$$\mathrm{Var}\,\widehat{\mathscr{F}}^{\mathrm{ort}}_{M}(\mathbf{z}) \leq \mathrm{Var}\,\widehat{\mathscr{F}}^{\mathrm{iid}}_{M}(\mathbf{z}) - \frac{2(d-1)}{M(d+2)}(\mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z}) - \mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{0}_{d}))^{2}.$$

As before, a block-orthogonal estimator leads to better concentration results. Theorem 6 follows directly from Remark 1 and Theorem 11. Theorem 8 follows directly from Remark 1, Theorems 9, 10 and a trivial observation that $\mathbb{E}\|\boldsymbol{\omega}\|^2 = d$ when $\boldsymbol{\omega} \sim \Omega$. Consequently, we only provide proofs of Theorems 10, 11 in the appendix.

4.5 Injecting input data priors through masking

In this section we consider an intriguing extension of GA and FAVOR+ which supports data-specific priors in the definition of self-attention while maintaining subquadratic computational complexity.

4.5.1 The definition of masked self-attention

Let $g^{(1)}, g^{(2)} : \mathbb{R}^d \to \mathbb{R}_+^M$ be functions from GA definition (Section 3.6) but mapping into positive-valued vectors for stability as discussed in Section 4.1. Note that FAVOR+ can be thought as a special case of GA with stochastic $g^{(1)}, g^{(2)}$ defined as

$$g^{(1)}_{\text{FAVOR+}}(\mathbf{x}) = M^{-1/2}\exp(d^{-1/2}\|\mathbf{x}\|/2)(f^{(1)}_{\text{pos}}(\boldsymbol{\omega}^{(m)}, d^{-1/4}\mathbf{x}))_{m=1}^M,$$
$$g^{(2)}_{\text{FAVOR+}}(\mathbf{y}) = M^{-1/2}\exp(d^{-1/2}\|\mathbf{y}\|/2)(f^{(2)}_{\text{pos}}(\boldsymbol{\omega}^{(m)}, d^{-1/4}\mathbf{y}))_{m=1}^M$$

where $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(M}$ are sampled according to Algorithm 2. Hence, we can discuss the case of GA without loss of generality.

Define *masked self-attention* as follows:

$$\text{Att}^{\text{MGA}}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = (\mathbf{D}^{\text{MGA}})^{-1}\mathbf{A}^{\text{MGA}}\mathbf{V} \in \mathbb{R}^{L\times d}, \quad \mathbf{A}^{\text{MGA}} = \mathbf{M} \circ \mathbf{A}^{\text{GA}}, \tag{4.11}$$
$$\mathbf{D}^{\text{MGA}} = \text{diag}(\mathbf{A}^{\text{MGA}}\mathbf{1}_L) \tag{4.12}$$

where $\mathbf{A}^{\text{GA}}$ is defined as in Section 3.6: $\mathbf{A}^{\text{GA}} = (g^{(1)}(\mathbf{Q}_i)^\top g^{(2)}(\mathbf{K}_j))_{i,j=1}^{L,L}$. $\mathbf{M} \in \mathbb{R}_+^{L\times L}$ is a *self-attention fmask* with non-negative entries. Both noncausal and causal GA (Section 3.6) are special cases of the definition (4.11-4.12). Indeed, $\mathbf{M} = \mathbf{1}_{L\times L}$ corresponds to noncausal GA and $\mathbf{M} = \text{tril}(\mathbf{1}_{L\times L})$ to causal GA. $\mathbf{M}$ can be used to inject prior knowledge about the input data into the model. For example, if the input has a graph structure, $\mathbf{M}$ can be an adjacency matrix of that graph, so that the information in self-attention is propagated along the edges of the graph. If, on the other hand, the input has a grid structure, $\mathbf{M}$ can define the weight of "interaction" between different nodes of the grid, where the weight of interaction only depends on the relative position between the two nodes. The latter case is discussed in detail in Section 4.5.3.

4.5.2 Efficient computation of masked self-attention

We next show that (4.11-4.12) can be evaluated efficiently when $M \ll L$ and $\mathbf{M}$ can be applied efficiently as a linear map. Observe that computing (4.11-4.12) reduces to finding $\mathbf{A}^{\mathrm{MGA}}\mathbf{C}$ where $\mathbf{C} = \begin{bmatrix}\mathbf{V} & \mathbf{1}_L\end{bmatrix}$. As shown in Section 3.6, $\mathbf{A}^{\mathrm{GA}}$ is a low-rank matrix, meaning that it can be expressed as $\mathbf{A}^{\mathrm{GA}} = \mathbf{P}^{(\mathrm{GA},1)}(\mathbf{P}^{(\mathrm{GA},2)})^\top$ where $\mathbf{P}^{(\mathrm{GA},1)}, \mathbf{P}^{(\mathrm{GA},2)} \in \mathbb{R}^{L\times M}$ are defined in (3.9). Hence, for $\mathbf{A}^{\mathrm{MGA}}$ we have:

$$\begin{aligned}\mathbf{A}^{\mathrm{MGA}} &= \mathbf{M} \circ \left(\mathbf{P}^{(\mathrm{GA},1)}(\mathbf{P}^{(\mathrm{GA},2)})^\top\right) = \mathbf{M} \circ \left(\sum_{m=1}^{M} \mathbf{P}^{(\mathrm{GA},1)}_{:,m}(\mathbf{P}^{(\mathrm{GA},2)}_{:,m})^\top\right) \\ &= \sum_{m=1}^{M}\left(\mathbf{M} \circ \left(\mathbf{P}^{(\mathrm{GA},1)}_{:,m}(\mathbf{P}^{(\mathrm{GA},2)}_{:,m})^\top\right)\right).\end{aligned}$$

Since (i,j)'th element of $\mathbf{M} \circ \left(\mathbf{P}^{(\mathrm{GA},1)}_{:,m}(\mathbf{P}^{(\mathrm{GA},2)}_{:,m})^\top\right)$ is $\mathbf{M}_{i,j}\mathbf{P}^{(\mathrm{GA},1)}_{i,m}\mathbf{P}^{(\mathrm{GA},2)}_{j,m}$, we conclude that, for all $1 \leq m \leq M$,

$$\mathbf{M} \circ \left(\mathbf{P}^{(\mathrm{GA},1)}_{:,m}(\mathbf{P}^{(\mathrm{GA},2)}_{:,m})^\top\right) = \mathrm{diag}(\mathbf{P}^{(\mathrm{GA},1)}_{:,m}) \times \mathbf{M} \times \mathrm{diag}(\mathbf{P}^{(\mathrm{GA},2)}_{:,m})$$

and, finally,

$$\begin{aligned}\mathbf{A}^{\mathrm{MGA}}\mathbf{C} &= \sum_{m=1}^{M}\left(\mathrm{diag}(\mathbf{P}^{(\mathrm{GA},1)}_{:,m}) \times \mathbf{M} \times \mathrm{diag}(\mathbf{P}^{(\mathrm{GA},2)}_{:,m})\right)\mathbf{C} \\ &= \sum_{m=1}^{M}\left(\mathrm{diag}(\mathbf{P}^{(\mathrm{GA},1)}_{:,m}) \overset{3}{\times} \mathbf{M} \overset{2}{\times} \mathrm{diag}(\mathbf{P}^{(\mathrm{GA},2)}_{:,m}) \overset{1}{\times} \mathbf{C}\right)\end{aligned}$$

where the numbers above "$\times$" define the order of efficient computations. For each $1 \leq m \leq M$, the first product takes $O(Ld)$ computations, the second one $dT_{\mathbf{M}}$ computations where $T_{\mathbf{M}}$ denotes the number of computations required to apply $\mathbf{M}$ as a linear map to a single vector. The third product takes $O(Ld)$ computations again. The final computational complexity of finding $\mathbf{A}^{\mathrm{MGA}}\mathbf{C}$ is, therefore, $O(MdT_{\mathbf{M}} + LMd)$ where we also assume that computing $\mathbf{P}^{(\mathrm{GA},1)}, \mathbf{P}^{(\mathrm{GA},2)}$ takes $O(LMd)$. This is usually the case, since, for FAVOR+ or GA, computing $g^{(1)}, g^{(2)}$ takes $O(Md)$ time. If $T_{\mathbf{M}}$ grows slower than $O(L^2)$ as the size of $\mathbf{M}$ increases (i.e. applying $\mathbf{M}$ as a linear map doesn't require materialization of $\mathbf{M} \in \mathbb{R}^{L\times L}$), the total complexity of the algorithm is much smaller than $O(L^2 d)$, i.e. the algorithm has a sub-quadratic complexity in L.

Algorithm 4 summarizes the routine for computing masked self-attention. Note that, as long as $\mathbf{M}$ has nonnegative entries, we shouldn't get numerical instabilities, discussed in Section 4.1, in the last line of the algorithm since the denominator has nonnegative entries.

In the next section, we discuss a particular instantiation of $\mathbf{M}$ which can be used with grid data like images, and is compatible with sub-quadratic masked self-attention.

Algorithm 4 Outline of the efficient masked self-attention algorithm.

Input: $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L \times d}$.
Output: Masked self-attention (4.11-4.12).
Compute $\mathbf{P}^{(1,\mathrm{GA})}, \mathbf{P}^{(2,\mathrm{GA})}$ according to (3.9);
Set $\mathbf{C} = \begin{bmatrix} \mathbf{V} & \mathbf{1}_L \end{bmatrix}$;
for $m \leftarrow 1$ to M **do**
    Compute $\mathbf{Buf}^{(1,m)} = \mathrm{diag}(\mathbf{P}^{(\mathrm{GA},2)}_{:,m}) \times \mathbf{C}$;
    Compute $\mathbf{Buf}^{(1,m)} = \mathbf{M} \times \mathbf{Buf}^{(1,m)}$;
    Compute $\mathbf{Buf}^{(1,m)} = \mathrm{diag}(\mathbf{P}^{(\mathrm{GA},1)}_{:,m}) \times \mathbf{Buf}^{(1,m)}$;
end for
Compute $\mathbf{Buf}^{(2)} = \sum_{m=1}^{M} \mathbf{Buf}^{(1,m)}$;
Return $\mathrm{diag}(\mathbf{Buf}^{(2)}_{:,d+1})^{-1} \mathbf{Buf}^{(2)}_{:,:d}$;

4.5.3 Multilevel Toeplitz masks and relative positional encoding

In this section we consider a design of $\mathbf{M}$ as the multilevel Toeplitz matrix. A remarkable property of such matrices is that they are compatible with the efficient matrix-vector product. We first define these matrices and then discuss the interpretations of these masks as relative positional encodings.

Definition 3 (Toeplitz matrices)**.** *We say that a matrix $\mathbf{M} \in \mathbb{R}^{L \times L}$ is Toeplitz (or 1-level block-Toeplitz) if, for all $1 \le i_1, i_2, j_1, j_2 \le L$, $\mathbf{M}_{i_1,j_1} = \mathbf{M}_{i_2,j_2}$ when $i_1 - j_1 = i_2 - j_2$.*

Definition 4 (u-level block-Toeplitz matrices)**.** *We say that $\mathbf{M} \in \mathbb{R}^{L \times L}$ is u-level block-Toeplitz for $u \ge 2$ if $\mathbf{M}$ consists of $L_u \times L_u$ blocks $\mathbf{M}^{(i,j)}$ of size $(L/L_u) \times (L/L_u)$ which are all $(u-1)$-level block-Toeplitz matrices and, for all $1 \le i_1, i_2, j_1, j_2 \le L_u$, $\mathbf{M}^{(i_1,j_1)} = \mathbf{M}^{(i_2,j_2)}$ when $i_1 - j_1 = i_2 - j_2$.*

Imagine a one-dimensional grid with a row of L nodes (Figure 4.2-I). Consider an arbitrary function $\psi_{1\mathrm{d}} : \{-L+1, \ldots, L-1\} \to \mathbb{R}_+$ and define a matrix $\mathbf{M} \in \mathbb{R}^{L \times L}$ as $\mathbf{M}_{i,j} = \psi_{1\mathrm{d}}(i-j)$. Then, according to Definition 3, $\mathbf{M}$ is a Toeplitz matrix. If the input data has a 1-d grid structure, e.g. it's a sequence, using such $\mathbf{M}$ as a mask in (4.11-4.12) would weigh self-attention coefficients according to a (signed) distance between pairs of elements, and the weights are defined by $2L-1$ values of the function ψ. Such masking is what we refer to as *1-d relative positional encoding* (Luo et al., 2021).

Next, imagine a two-dimensional grid of size $L_1 \times L_2$, $L = L_1 L_2$, where nodes are labeled as (i', i''), $1 \leq i' \leq L_1, 1 \leq i'' \leq L_2$, according to their position in the grid (for instance, a horizontal and vertical index). Number the nodes from 1 to L according to the reflected lexicographic order of (i', i''), i.e. $(i', i'') < (j', j'')$ if $i' < j' \vee (i' = j' \wedge i'' < j'')$. Consider a function $\psi_{2\text{d}} : \{-L_1 + 1, \ldots, L_1 - 1\} \times \{-L_2 + 1, \ldots, L_2 - 1\} \to \mathbb{R}_+$ and define a matrix $\mathbf{M} \in \mathbb{R}^{L \times L}$ as $\mathbf{M}_{i,j} = \psi_{2\text{d}}(i' - j', i'' - j'')$ where (i', i''), (j', j'') are labels of nodes with numbers i, j respectively. Then, $\mathbf{M}$ is a 2-level block-Toeplitz matrix since it can be split into blocks $\mathbf{M}^{(i',j')}$ of size $L_2 \times L_2$ where $\mathbf{M}^{(i',j')}$ corresponds to a fixed pair of $1 \leq i', j' \leq L_1$. $\mathbf{M}^{(i',j')}$ is a Toeplitz matrix since $\mathbf{M}^{(i',j')}_{i'',j''} = \psi_{2\text{d}}(\mathscr{D}, i'' - j'')$ where $\mathscr{D} = i' - j'$ is fixed for $\mathbf{M}^{(i',j')}$. Further, since $\mathbf{M}^{(i',j')}$ only depends on the difference $\mathscr{D} = i' - j'$, we conclude that $\mathbf{M}$ satisfies Definition 4.

If the input data has a 2-d grid structure, e.g. it's an image, taking such 2-level block-Toeplitz matrix as a mask in (4.11-4.12) would weigh self-attention coefficients according to $\psi_{2\text{d}}$ which is shift-invariant and depends on relative positions between nodes in the grid. We refer to such masking mechanism as *2-d relative positional encoding*. Analogously, *u-d relative positional encodings* can be defined on u-dimensional masks (see Figure 4.2 for an illustration). A remarkable fact about L-sized u-level block-Toeplitz masks is that they support $O(L\log(L))$ matrix-vector multiplication via fast Fourier transform for any fixed u (Lee, 1986). This, given the Algorithm 4, results in a $O(L\log(L)Md)$ procedure for computing masked self-attention which is tractable for long sequences. We can leave the values of ψ_{ud} as trainable parameters with an appropriate reparametrization to keep them nonnegative ($z \to |z|$).

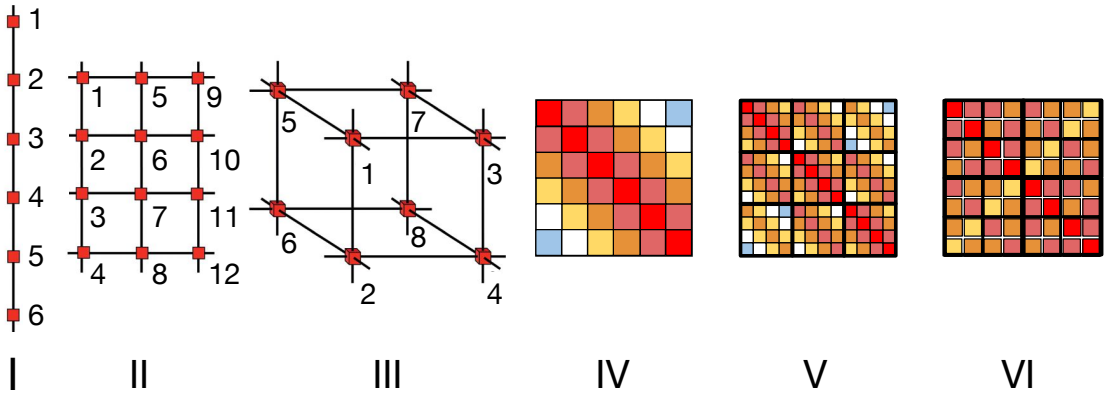


Fig. 4.2 An illustration of multilevel block-Toeplitz matrices. I, II, III – 1-d, 2-d and 3-d grids numbered according to the lexicographic order. IV, V, VI – corresponding 1-level, 2-level and 3-level block-Toeplitz matrices (colors indicate different values).

4.6 Experiments

We compare FAVOR+ with FAVOR and evaluate masked self-attention in two benchmarks: masked language modelling on PG-19 dataset and image recognition on ImageNet.

4.6.1 Masked language modelling on text

The PG-19 dataset compr is presented as a challenging long range text modeling task. It consists of out-of-copyright Project Gutenberg books published before 1919. It does not have a fixed vocabulary size, instead opting for any tokenization which can model an arbitrary string of text. We use a unigram SentencePiece vocabulary sentencepiece with 32768 tokens, which maintains whitespace and is completely invertible to the original book text. For the task of masked language modelling on PG-19, we train Transformer and its efficient variants using "Regular" configuration from Chapter 3: $(h, s, d_{\text{ff}}, d_{\text{hid}}) = (8, 6, 2048, 512)$. We subsample sequences to match the length $L = 1024$. We reuse hyperparameters from Appendix 3.B and compare the vanilla Transformer, different Performer variants and Linformer (Section 3.7). For Performer, we compare two strategies for handling the random block-orthogonal matrix $\mathbf{\Omega}$: either to draw it once and reuse during training ("no redraw"), or redraw it for every forward pass ("redraw"). Same applies for the random projection matrix in Linformer. Further, we also evaluate RegRFs (Section 4.3) by using them in Performer instead of FAVOR+. We use perplexity as a metric for comparison (Section 3.8.3). Figure 4.3 demonstrates the results. As we observe, this setup is an example where FAVOR demonstrates an unstable behaviour and completely fails to train. At the same time, all Performer variants have a very smooth training curve, with "redraw" variants eventually matching the Transformer baseline which is not the case for the Linformer. We conclude that FAVOR+ is indeed a better default choice for a practitioner.

4.6.2 Masked self-attention for image recognition

We evaluate masked self-attention with 2-level block-Toeplitz masks on the task of image classification with ViT (Section 2.2.5) using Performer backbone. We take ImageNet dataset (Deng et al., 2009) and downsample images to the size of 224×224. Then, we use a patch size of 14×14 resulting in the sequences of length $L = 197$ (16×16 plus an extra class token). We train our models on 16 TPUs with a batch size of 256 per TPU. For the mask, we take a learnable 2-level block-Toeplitz mask which is initialized at ones and corresponds to the 16×16 grid (and a fixed weight of one for all interactions with the class token). We train Performers with several GA variants: Performer-ReLU, Performer-ELU, Performer-SQR

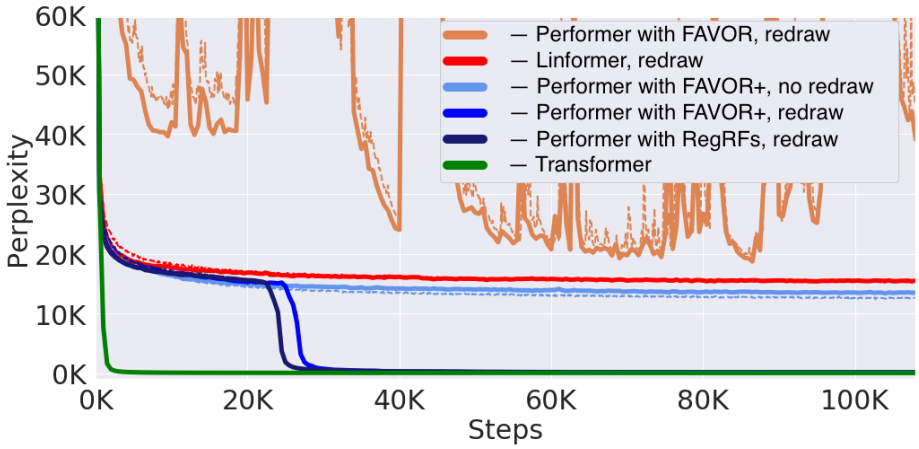


Fig. 4.3 Masked language modelling on PG-19 benchmark. Dashed and solid lines indicate train and validation metric respectively.

(see Section 3.6). We compare masked variants and non-masked variants in Figure 4.4. We observe a consistent improvement $(2.5 - 3.4\%)$ achieved by using masked self-attention which proves that injecting input data priors through masking can be very beneficial for Performer training.

4.7 Discussion

In this chapter we introduced FAVOR+ mechanism (*F*ast *A*ttention *V*ia *positive* *O*rthogonal *R*andom features): an extension of FAVOR mechanism for random feature approximation of self-attention. Positivity of FAVOR+'s approximation leads to a numerically stable training since the self-attention denominator is positive and not too small. We performed a theoretical analysis of FAVOR+ and 1) showed that positive random features benefit from orthogonality and 2) proved right tail concentration results for the regularized variant of positive random features. Further, we introduced an extension of Performer via the masked self-attention which allows injection of input data driven priors into the self-attention weights while still maintaining sub-quadratic computational complexity. We demonstrated efficiency of FAVOR+ and masked self-attention in real life large scale training scenarios: masked language modelling and image recognition respectively.

In the next chapter, we will present yet another extension of Performers for low-memory training. And, in Chapter 6, we will extend FAVOR+ to a more precise but still positive random feature mechanism with explicit exponential concentration and better empirical performance.

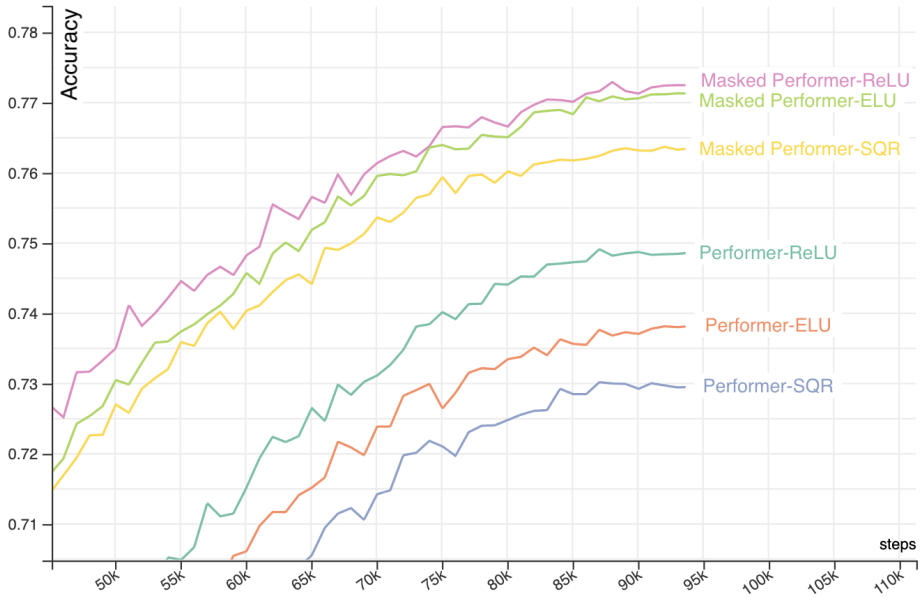


Fig. 4.4 Image recognition using Performers with masked self-attention ("Masked Performer-...") and without any masking ("Performer-..."). We don't report regular ViT since it has a quadratic self-attention complexity. Validation accuracy (%).

Appendix 4.A Proofs

Proof of Theorem 5

Proof. We first deduce that for any $\mathbf{x},\mathbf{y} \in \mathbb{R}^d$

$$K(\mathbf{x},\mathbf{y}) = \exp(-\|\mathbf{x}\|^2)\exp(\|\mathbf{x}+\mathbf{y}\|^2/2)\exp(-\|\mathbf{y}\|^2). \tag{4.13}$$

Next, let $\mathbf{w} \in \mathbb{R}^d$. We use the fact that

$$1 = (2\pi)^{-d/2}\int \exp(-\|\boldsymbol{\omega}-\mathbf{c}\|^2/2)\mathrm{d}\boldsymbol{\omega} \tag{4.14}$$

for any $\mathbf{c} \in \mathbb{R}^d$. We set $\mathbf{c} = \mathbf{x}+\mathbf{y}$ and multiply (4.14) by $\exp(\|\mathbf{x}+\mathbf{y}\|^2/2)$:

$$\begin{aligned}
\exp(\|\mathbf{x}+\mathbf{y}\|^2/2) &= (2\pi)^{-d/2}\exp(\|\mathbf{x}+\mathbf{y}\|^2/2)\int \exp(-\|\boldsymbol{\omega}-(\mathbf{x}+\mathbf{y})\|^2/2)\mathrm{d}\boldsymbol{\omega} \\
&= (2\pi)^{-d/2}\int \exp(-\|\boldsymbol{\omega}\|^2/2+\boldsymbol{\omega}^\top(\mathbf{x}+\mathbf{y})-\|\mathbf{x}+\mathbf{y}\|^2/2+\|\mathbf{x}+\mathbf{y}\|^2/2)\mathrm{d}\boldsymbol{\omega} \\
&= (2\pi)^{-d/2}\int \exp(-\|\boldsymbol{\omega}\|^2/2+\boldsymbol{\omega}^\top(\mathbf{x}+\mathbf{y}))\mathrm{d}\boldsymbol{\omega} \\
&= (2\pi)^{-d/2}\int \exp(-\|\boldsymbol{\omega}\|^2/2)\exp(\boldsymbol{\omega}^\top\mathbf{x})\exp(\boldsymbol{\omega}^\top\mathbf{y})d\boldsymbol{\omega}
\end{aligned}$$

$$= \mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}\left(\exp(\boldsymbol{\omega}^\top \mathbf{x})\exp(\boldsymbol{\omega}^\top \mathbf{y})\right) \tag{4.15}$$

where we perform a number of straightforward transitions. The last transition is where we express the integral as an expectation with respect to the standard Gaussian distribution with the density function $p_{\mathrm{SG}}(\boldsymbol{\omega}) = (2\pi)^{-d/2}\exp(-\|\boldsymbol{\omega}\|^2)$. Combining (4.13) and (4.15), we finally deduce that

$$K(\mathbf{x},\mathbf{y}) = \mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}\left(\exp(\boldsymbol{\omega}^\top \mathbf{x} - \|\mathbf{x}\|^2)\exp(\boldsymbol{\omega}^\top \mathbf{y} - \|\mathbf{y}\|^2)\right)$$

which is equivalent to (4.3). The second part of the proof establishes the expression for the variance. We have:

$$\begin{aligned}\mathrm{Var}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}\left(f^{(1)}_{\mathrm{pos}}(\boldsymbol{\omega},\mathbf{x})f^{(2)}_{\mathrm{pos}}(\boldsymbol{\omega},\mathbf{y})\right) &= \mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}\left(f^{(1)}_{\mathrm{pos}}(\boldsymbol{\omega},\mathbf{x})f^{(2)}_{\mathrm{pos}}(\boldsymbol{\omega},\mathbf{y})\right)^2 \\ &- \left(\mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}f^{(1)}_{\mathrm{pos}}(\boldsymbol{\omega},\mathbf{x})f^{(2)}_{\mathrm{pos}}(\boldsymbol{\omega},\mathbf{y})\right)^2 \\ &= \mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}\left(f^{(1)}_{\mathrm{pos}}(\boldsymbol{\omega},\mathbf{x})f^{(2)}_{\mathrm{pos}}(\boldsymbol{\omega},\mathbf{y})\right)^2 - K(\mathbf{x},\mathbf{y})^2. \end{aligned} \tag{4.16}$$

Next, by the definition of $f^{(1)}_{\mathrm{pos}}, f^{(2)}_{\mathrm{pos}}$, we have:

$$\begin{aligned}\mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}\left(f^{(1)}_{\mathrm{pos}}(\boldsymbol{\omega},\mathbf{x})f^{(2)}_{\mathrm{pos}}(\boldsymbol{\omega},\mathbf{y})\right)^2 &= \mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}\left(\exp(2\boldsymbol{\omega}^\top \mathbf{x} - 2\|\mathbf{x}\|^2)\exp(2\boldsymbol{\omega}^\top \mathbf{y} - 2\|\mathbf{y}\|^2)\right) \\ &= \exp(-2\|\mathbf{x}\|^2 - 2\|\mathbf{y}\|^2)\mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}\left(\exp(\boldsymbol{\omega}^\top (2\mathbf{x}))\exp(\boldsymbol{\omega}^\top (2\mathbf{y}))\right) \end{aligned} \tag{4.17}$$

By substituting $\mathbf{x} \rightarrow 2\mathbf{x}$ and $\mathbf{y} \rightarrow 2\mathbf{y}$ in (4.15) we deduce that

$$\mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}\left(\exp(\boldsymbol{\omega}^\top (2\mathbf{x}))\exp(\boldsymbol{\omega}^\top (2\mathbf{y}))\right) = \exp(\|2\mathbf{x}+2\mathbf{y}\|^2/2) = \exp(2\|\mathbf{x}+\mathbf{y}\|^2).$$

We use it in (4.17) and deduce that

$$\mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}\left(f^{(1)}_{\mathrm{pos}}(\boldsymbol{\omega},\mathbf{x})f^{(2)}_{\mathrm{pos}}(\boldsymbol{\omega},\mathbf{y})\right)^2 = \exp(-2\|\mathbf{x}\|^2 - 2\|\mathbf{y}\|^2)\exp(2\|\mathbf{x}+\mathbf{y}\|^2) = \exp(4\mathbf{x}^\top \mathbf{y}).$$

This together with (4.16) results in (4.4). □

Proof of Theorem 7

Proof. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. Denote $\mathbf{z} = \mathbf{x} + \mathbf{y}$. We use $K^{\mathrm{reg}}(\mathbf{x}, \mathbf{y})$'s definition and Taylor expansion of $\exp(\cdot)$ to deduce that

$$\begin{aligned} K^{\mathrm{reg}}(\mathbf{x}, \mathbf{y}) &= \exp(-\|\mathbf{x}\|^2 - \|\mathbf{y}\|^2)\mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})} \exp(\frac{\sqrt{d}}{\|\boldsymbol{\omega}\|}\boldsymbol{\omega}^\top \mathbf{z}) \\ &= \exp(-\|\mathbf{x}\|^2 - \|\mathbf{y}\|^2) \sum_{k=0}^{\infty} \frac{1}{k!} \mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})} \left(\frac{\sqrt{d}}{\|\boldsymbol{\omega}\|}\boldsymbol{\omega}^\top \mathbf{z} \right)^k. \end{aligned}$$

Observe that since $p_{\mathrm{SG}}(\cdot)$ is an isotropic distribution, we can 1) zero out odd terms in the sum and 2) replace $\mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})} \left(\frac{\sqrt{d}}{\|\boldsymbol{\omega}\|}\boldsymbol{\omega}^\top \mathbf{z} \right)^k$ with $\mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})} \left(\frac{\sqrt{d}\|\mathbf{z}\|}{\|\boldsymbol{\omega}\|}\boldsymbol{\omega}^\top \mathbf{e}_1 \right)^k$. As the result, we have:

$$K^{\mathrm{reg}}(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2) \sum_{k=0}^{\infty} \frac{1}{(2k)!} \|\mathbf{z}\|^{2k} d^k \mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})} \left(\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}\mathbf{e}_1 \right)^{2k}. \tag{4.18}$$

Let us denote: $\mathscr{A}(k,d) = \mathbb{E}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}(\frac{\omega}{\|\omega\|}\mathbf{e}_1)^{2k}$. It turns out that, if $d \geq 2$,

$$\mathscr{A}(2k,d) = \frac{(2k-1)!!}{(d+2k-2)(d+2k-4)\ldots d} \tag{4.19}$$

where $(2k-1)!!$ denotes a double factorial: $(2k-1)!! = (2k-1)(2k-3)\ldots 1$. The proof of this fact can be found in the supplement of geom. Applying the above formula, we get:

$$\begin{aligned} K^{\mathrm{reg}}(\mathbf{x}, \mathbf{y}) &= \exp(-\|\mathbf{x}\|^2 - \|\mathbf{y}\|^2) \sum_{k=0}^{\infty} \frac{1}{(2k)!} \|\mathbf{z}\|^{2k} d^k \frac{(2k-1)!!}{(d+2k-2)(d+2k-4)\ldots d} \\ &= \exp(-\|\mathbf{x}\|^2 - \|\mathbf{y}\|^2) \sum_{k=0}^{\infty} \frac{w^k}{k!} \Xi(k,d) \end{aligned}$$

where $w = \frac{\|\mathbf{z}\|^2}{2}$ and $\Xi(k,d) = \frac{d^k}{(d+2k-2)(d+2k-4)\ldots d}$.

Therefore, we obtain:

$$\frac{K^{\mathrm{reg}}(\mathbf{x}, \mathbf{y})}{K(\mathbf{x}, \mathbf{y})} = e^{-w} \sum_{k=0}^{\infty} \frac{w^k}{k!} \Xi(k,d).$$

For $k \geq 0$ we have: $\Xi(k,d) \leq 1$, thus:

$$\frac{K^{\mathrm{reg}}(\mathbf{x}, \mathbf{y})}{K(\mathbf{x}, \mathbf{y})} \leq 1.$$

We also have:

$$\frac{K^{\mathrm{reg}}(\mathbf{x},\mathbf{y})}{K(\mathbf{x},\mathbf{y})} = e^{-w}\sum_{k=0}^{\lfloor d^{1/3}\rfloor}\frac{w^k}{k!}\Xi(k,d) + e^{-w}\sum_{k=\lfloor d^{1/3}\rfloor+1}^{\infty}\frac{w^k}{k!}\Xi(k,d)$$
$$\geq \Xi(\lfloor d^{1/3}\rfloor,d)e^{-w}\sum_{k=0}^{\lfloor d^{1/3}\rfloor}\frac{w^k}{k!} + e^{-w}\sum_{k=\lfloor d^{1/3}\rfloor+1}^{\infty}\frac{w^k}{k!}\Xi(k,d)$$
$$\geq \Xi(\lfloor d^{1/3}\rfloor,d)e^{-w}\sum_{k=0}^{\lfloor d^{1/3}\rfloor}\frac{w^k}{k!}$$

where we use $\Xi(\lfloor d^{1/3}\rfloor,d) \leq \Xi(k,d)$ for $k \leq \lfloor d^{1/3}\rfloor$. Next, we use the Taylor expansion $e^w = \sum_{k=0}^{\infty}\frac{w^k}{k!}$ and deduce:

$$\Xi(\lfloor d^{1/3}\rfloor,d)e^{-w}\sum_{k=0}^{\lfloor d^{1/3}\rfloor}\frac{w^k}{k!} = \Xi(\lfloor d^{1/3}\rfloor,d)\left(1 - e^{-w}\sum_{k=\lfloor d^{1/3}\rfloor+1}^{\infty}\frac{w^k}{k!}\right)$$
$$= \Xi(\lfloor d^{1/3}\rfloor,d)(1-\mathbb{P}(\mathrm{Po}(w) > d^{1/3})),$$

where $\mathrm{Po}(w)$ stands for the random variable of Poisson distribution with parameter w. Observe that, by the definition of $\Xi(k,d)$, for $k \geq 1$ we have:

$$\Xi(k,d) \geq \left(\frac{d}{d+2k-2}\right)^k \geq \left(1+\frac{2k-2}{d}\right)^k$$

where the second transition follows from $d^2 \geq d^2-(2k-2)^2 = (d-2k+2)(d+2k-2)$ from which we get $\frac{d}{d+2k-2} \geq \frac{d+2k-2}{d}$. Now, we set $k = \lfloor d^{1/3}\rfloor \geq 1$ and obtain

$$\frac{K^{\mathrm{reg}}(\mathbf{x},\mathbf{y})}{K(\mathbf{x},\mathbf{y})} \geq \left(1-\frac{2\lfloor d^{1/3}\rfloor-2}{d}\right)^{\lfloor d^{1/3}\rfloor}(1-\mathbb{P}(\mathrm{Po}(w) > d^{1/3})).$$

For d big enough we get $\left|\frac{2\lfloor d^{1/3}\rfloor-2}{d}\right| < 1$ which allows writing $\left(1-\frac{2\lfloor d^{1/3}\rfloor-2}{d}\right)^{\lfloor d^{1/3}\rfloor} = \exp\left(\lfloor d^{1/3}\rfloor\log\left(1-\frac{2\lfloor d^{1/3}\rfloor-2}{d}\right)\right)$ and use the Taylor expansion:

$$\log\left(1-\frac{2\lfloor d^{1/3}\rfloor-2}{d}\right) = \sum_{k=1}^{\infty}(-1)^k\frac{\left(\frac{2\lfloor d^{1/3}\rfloor-2}{d}\right)^k}{k}.$$

Notice that $w = \frac{\|\mathbf{z}\|^2}{2} = \frac{1}{2}(\|\mathbf{x}\|^2 + 2\mathbf{x}^\top\mathbf{y} + \|\mathbf{y}\|^2) \leq 2\mathscr{C}$. When $d > (2\mathscr{C})^3 \geq w^3$ we deduce for $t = \log\left(\frac{d^{1/3}}{w}\right) > 0$ that

$$\begin{aligned}\frac{K^{\text{reg}}(\mathbf{x},\mathbf{y})}{K(\mathbf{x},\mathbf{y})} &\geq \exp\left(\lfloor d^{1/3}\rfloor \sum_{k=1}^{\infty}(-1)^k \frac{\left(\frac{2\lfloor d^{1/3}\rfloor - 2}{d}\right)^k}{k}\right)(1-\mathbb{P}(\text{Po}(w) > d^{1/3})) \\ &\geq \exp\left(-\frac{2}{d^{\frac{1}{3}}} + o\left(\frac{1}{d^{\frac{1}{3}}}\right)\right)(1-\mathbb{P}(t\text{Po}(w) \geq td^{1/3})) \\ &= \exp\left(-\frac{2}{d^{\frac{1}{3}}} + o\left(\frac{1}{d^{\frac{1}{3}}}\right)\right)(1-\mathbb{P}(\exp(t\text{Po}(w) - td^{1/3}) \geq 1)).\end{aligned}$$

From Markov's inequality we deduce:

$$\mathbb{P}(\exp(t\text{Po}(w) - td^{1/3}) \geq 1) \leq \exp(-td^{1/3})\mathbb{E}(\exp(t\text{Po}(w))).$$

Further, the moment generating function formula for the Poisson distribution has a form:

$$\mathbb{E}[\exp(t\text{Po}(w))] = \exp(w(\exp(t)-1)) = \exp(d^{1/3} - w)$$

where we use the definition of t. Using this, we continue the chain of inequalities:

$$\begin{aligned}\frac{K^{\text{reg}}(\mathbf{x},\mathbf{y})}{K(\mathbf{x},\mathbf{y})} &\geq \exp\left(-\frac{2}{d^{\frac{1}{3}}} + o\left(\frac{1}{d^{\frac{1}{3}}}\right)\right)(1-\mathbb{P}(\exp(t\text{Po}(w) - td^{1/3}) \geq 1)) \\ &\geq \exp\left(-\frac{2}{d^{\frac{1}{3}}} + o\left(\frac{1}{d^{\frac{1}{3}}}\right)\right)(1-\exp(-td^{1/3})\mathbb{E}[\exp(t\text{Po}(w))]) \\ &= \exp\left(-\frac{2}{d^{\frac{1}{3}}} + o\left(\frac{1}{d^{\frac{1}{3}}}\right)\right)(1-\exp(-w - d^{1/3}(t-1))) \\ &= \left(1-\frac{2}{d^{\frac{1}{3}}} + o\left(\frac{1}{d^{\frac{1}{3}}}\right)\right)(1-\exp(-w - d^{1/3}(t-1))).\end{aligned}$$

By unwrapping the definition of t again and recalling that $w \leq 2\mathscr{C}$, we conclude that:

$$\frac{K^{\text{reg}}(\mathbf{x},\mathbf{y})}{K(\mathbf{x},\mathbf{y})} \geq \left(1-\frac{2}{d^{\frac{1}{3}}} + o\left(\frac{1}{d^{\frac{1}{3}}}\right)\right)\left(1-\exp(-2\mathscr{C})\left(\frac{d^{\frac{1}{3}}}{2e\cdot\mathscr{C}}\right)^{-d^{\frac{1}{3}}}\right) = 1-\frac{2}{d^{\frac{1}{3}}} + o\left(\frac{1}{d^{\frac{1}{3}}}\right).$$

That completes the proof. □

Proof of Theorem 9

Proof. For any $\phi > 0$ we have:

$$\mathbb{P}(\widehat{\mathscr{F}}^{\mathrm{iid}}_M(\mathbf{z}) > a) = \mathbb{P}\left(\sum_{m=1}^{M} \mathscr{G}((\boldsymbol{\omega}^{(\mathrm{iid},m)})^\top \mathbf{z}) > Ma\right)$$
$$= \mathbb{P}\left(\sum_{m=1}^{M} \phi\mathscr{G}((\boldsymbol{\omega}^{(\mathrm{iid},m)})^\top \mathbf{z}) > \phi Ma\right) = \mathbb{P}\left(\prod_{m=1}^{M} \exp\left(\phi\mathscr{G}((\boldsymbol{\omega}^{(\mathrm{iid},m)})^\top \mathbf{z})\right) > \exp(\phi Ma)\right).$$

We apply Markov's inequality and deduce that

$$\mathbb{P}\left(\prod_{m=1}^{M} \exp\left(\phi\mathscr{G}((\boldsymbol{\omega}^{(\mathrm{iid},m)})^\top \mathbf{z})\right) > \exp(\phi Ma)\right)$$
$$\leq \exp(-\phi Ma)\mathbb{E}\prod_{m=1}^{M} \exp\left(\phi\mathscr{G}((\boldsymbol{\omega}^{(\mathrm{iid},m)})^\top \mathbf{z})\right)$$
$$= \exp(-\phi Ma)\prod_{m=1}^{M} \mathbb{E}\exp\left(\phi\mathscr{G}((\boldsymbol{\omega}^{(\mathrm{iid},m)})^\top \mathbf{z})\right) = \exp(-\phi Ma)\left(\mathbb{E}\exp\left(\phi Z\right)\right)^M$$
$$= \exp(-\phi Ma)\mathscr{M}_Z(\phi)^M$$

where we can put expectation inside the product since $\mathscr{G}((\boldsymbol{\omega}^{(\mathrm{iid},1)})), \ldots, \mathscr{G}((\boldsymbol{\omega}^{(\mathrm{iid},M)}))$ are all independent. Taking all together, we have:

$$\mathbb{P}(\widehat{\mathscr{F}}^{\mathrm{iid}}_M(\mathbf{z}) > a) \leq \exp(-\phi Ma)\mathscr{M}_Z(\phi)^M.$$

□

Proof of Theorem 10

Proof. Note that by the analogous application of Markov's inequality as in Theorem 9, we get:

$$\mathbb{P}(\widehat{\mathscr{F}}^{\mathrm{ort}}_M(\mathbf{z})) > a) \leq \frac{\mathbb{E}(\exp(\phi(Z^{\mathrm{ort}}_1 + \ldots + Z^{\mathrm{ort}}_M)))}{\exp(\phi Ma)}$$

where we have: $Z^{\mathrm{ort}}_m = g((\boldsymbol{\omega}^{(\mathrm{ort},m)})^\top \mathbf{z})$. We see that it suffices to show that for any $\phi > 0$ the following holds: $\mathbb{E}(\exp(\phi(Z^{\mathrm{ort}}_1 + \ldots + Z^{\mathrm{ort}}_M))) < \mathbb{E}(\exp(\phi(Z^{\mathrm{iid}}_1 + \ldots + Z^{\mathrm{iid}}_M)))$ where $Z^{\mathrm{iid}}_m =$

$g((\boldsymbol{\omega}^{(\mathrm{iid},m)})^\top \mathbf{z})$. We have:

$$\mathbb{E}(\exp(\phi(Z_1^{\mathrm{ort}}+\cdots+Z_M^{\mathrm{ort}}))) = \mathbb{E}\left(\sum_{k=0}^{\infty}\frac{(\phi\sum_{m=1}^{M}Z_m^{\mathrm{ort}})^k}{k!}\right) = \mathbb{E}\left(\sum_{k=0}^{\infty}\frac{\phi^k}{k!}\left(\sum_{m=1}^{M}Z_m^{\mathrm{ort}}\right)^k\right) =$$

$$\sum_{k=0}^{\infty}\frac{\phi^k}{k!}\mathbb{E}\left(\left(\sum_{m=1}^{M}Z_m^{\mathrm{ort}}\right)^k\right) = \sum_{k=0}^{\infty}\frac{\phi^k}{k!}\mathbb{E}\left(\sum_{(k_1,\ldots,k_M)\in\mathscr{S}_k}\binom{k}{k_1,\ldots,k_M}(Z_1^{\mathrm{ort}})^{k_1}\ldots(Z_M^{\mathrm{ort}})^{k_M}\right),$$

where $\mathscr{S}_k = \{(k_1,\ldots,k_M)\in\mathbb{Z}^d : k_1,\ldots,k_M\geq 0, k_1+\cdots+k_M=k\}$.

Thus we have:

$$\mathbb{E}(\exp(\phi(Z_1^{\mathrm{ort}}+\cdots+Z_M^{\mathrm{ort}}))) = \sum_{k=0}^{\infty}\frac{\phi^k}{k!}\sum_{(k_1,\ldots,k_M)\in\mathscr{S}_k}\binom{k}{k_1,\ldots,k_M}\mathbb{E}((Z_1^{\mathrm{ort}})^{k_1}\ldots(Z_M^{\mathrm{ort}})^{k_M}).$$

Similarly, we get:

$$\mathbb{E}(\exp(\phi(Z_1^{\mathrm{iid}}+\cdots+Z_M^{\mathrm{iid}}))) = \sum_{k=0}^{\infty}\frac{\phi^k}{k!}\sum_{(k_1,\ldots,k_M)\in\mathscr{S}_k}\binom{k}{k_1,\ldots,k_M}\mathbb{E}((Z_1^{\mathrm{iid}})^{k_1}\ldots(Z_M^{\mathrm{iid}})^{k_M}).$$

Therefore we get:

$$\Delta = \mathbb{E}(\exp(\phi(Z_1^{\mathrm{iid}}+\cdots+Z_M^{\mathrm{iid}}))) - \mathbb{E}(\exp(\phi(Z_1^{\mathrm{ort}}+\cdots+Z_M^{\mathrm{ort}})))$$
$$= \sum_{k=0}^{\infty}\frac{\phi^k}{k!}\sum_{(k_1,\ldots,k_M)\in\mathscr{S}_k}\binom{k}{k_1,\ldots,k_M}\left(\mathbb{E}((Z_1^{\mathrm{iid}})^{k_1}\ldots(Z_M^{\mathrm{iid}})^{k_M}) - \mathbb{E}((Z_1^{\mathrm{ort}})^{k_1}\ldots(Z_M^{\mathrm{ort}})^{k_M})\right)$$

Using the power series of $\mathscr{G}$, we can rewrite each Z_m^{ort} as:

$$Z_m^{\mathrm{ort}} = \sum_{k'=0}^{\infty} a_{k'}((\boldsymbol{\omega}_m^{\mathrm{ort}})^\top \mathbf{z})^{k'} \tag{4.20}$$

where $\mathscr{F}_{\Omega,\mathscr{G}}(z) = \sum_{k'=0}^{\infty} a_{k'} z^{k'}$ and $a_0, a_1, \ldots \geq 0$. Similarly,

$$Z_m^{\mathrm{iid}} = \sum_{k'=0}^{\infty} a_{k'}((\boldsymbol{\omega}_m^{\mathrm{iid}})^\top \mathbf{z})^{k'}. \tag{4.21}$$

By plugging expressions for Z_m^{ort} and Z_m^{iid} into the formula for Δ and expanding power expressions, we obtain:

$$\Delta = \sum_{k=0}^{\infty} \frac{\phi^k}{k!} \sum_{(k_1,\ldots,k_M)\in\mathscr{S}_k} \binom{k}{k_1,\ldots,k_M} \sum_{(d_1,\ldots,d_M)\in\mathscr{D}(k_1,\ldots,k_M)} \widehat{c}_{j_1,\ldots,j_M}(d_1,\ldots,d_M) \times \widehat{\Delta}(d_1,\ldots,d_M),$$

for some $\mathscr{D}(k_1,\ldots,k_M) \subseteq (\{0\}\cup\mathbb{N})^M$ and some nonnegative $\widehat{c}_{j_1,\ldots,j_M}(d_1,\ldots,d_M)$ (exact formula for those can be given but we do not need it) and $\widehat{\Delta}(d_1,\ldots,d_M)$ defined as:

$$\widehat{\Delta}(d_1,\ldots,d_M) = \mathbb{E}(((\boldsymbol{\omega}^{(\mathrm{iid},1)})^\top \mathbf{z})^{d_1} \ldots ((\boldsymbol{\omega}^{(\mathrm{iid},M)})^\top \mathbf{z})^{d_M}) - \mathbb{E}(((\boldsymbol{\omega}^{(\mathrm{ort},1)})^\top \mathbf{z})^{d_1} \ldots ((\boldsymbol{\omega}^{(\mathrm{ort},M)})^\top \mathbf{z})^{d_M}). \tag{4.22}$$

Our next goal is to rewrite the formula for $\widehat{\Delta}(d_1,\ldots,d_M)$. Denote:

$$Y = ((\boldsymbol{\omega}^{(\mathrm{ort},1)})^\top \mathbf{z})^{d_1} \ldots ((\boldsymbol{\omega}^{(\mathrm{ort},M)})^\top \mathbf{z})^{d_M}.$$

Observe that Y has the same distribution as Y' defined as:

$$Y' = (\mathbf{e}_1^\top \frac{\mathbf{g}}{\|\mathbf{g}\|}\|\mathbf{z}\|)^{d_1} \ldots (\mathbf{e}_M^\top \frac{\mathbf{g}}{\|\mathbf{g}\|}\|\mathbf{z}\|)^{d_M} (\|\boldsymbol{\omega}^{(\mathrm{ort},1)}\|)^{d_1} \ldots (\|\boldsymbol{\omega}^{(\mathrm{ort},M)}\|)^{d_M},$$

where $\mathbf{g}$ is a vector taken from the $\mathscr{N}(0,1)^d$ distribution independently from $\boldsymbol{\omega}^{(\mathrm{ort},1)}$, ..., $\boldsymbol{\omega}^{(\mathrm{ort},M)}$.

This comes from the fact that for a fixed $\mathbf{z}$ one can think about the set: $\frac{\boldsymbol{\omega}^{(\mathrm{ort},1)}}{\|\boldsymbol{\omega}^{(\mathrm{ort},1)}\|}, \ldots, \frac{\boldsymbol{\omega}^{(\mathrm{ort},M)}}{\|\boldsymbol{\omega}^{(\mathrm{ort},M)}\|}$ as a random rotation of the system of M canonical basis vectors: $\mathbf{e}_1,\ldots,\mathbf{e}_M$. Thus, instead of applying a random rotation to $\mathbf{e}_1,\ldots,\mathbf{e}_M$, one can equivalently randomly rotate vector $\mathbf{z}$. Randomly rotated vector $\mathbf{z}$ has the same distribution as: $\frac{\mathbf{g}}{\|\mathbf{g}\|}\|\mathbf{z}\|$.

Now note that lengths of vectors $\boldsymbol{\omega}^{(\mathrm{ort},1)},\ldots,\boldsymbol{\omega}^{(\mathrm{ort},M)}$ are chosen independently. Therefore, we obtain:

$$\mathbb{E}(((\boldsymbol{\omega}^{(\mathrm{ort},1)})^\top \mathbf{z})^{d_1} \ldots ((\boldsymbol{\omega}^{(\mathrm{ort},M)})^\top \mathbf{z})^{d_M}) = \mathbb{E}((\|\boldsymbol{\omega}^{(\mathrm{ort},1)}\|)^{d_1}) \ldots \mathbb{E}((\|\boldsymbol{\omega}^{(\mathrm{ort},M)}\|)^{d_M}) \mathbb{E}((\mathbf{e}_1^\top \mathbf{v})^{d_1} \ldots (\mathbf{e}_M^\top \mathbf{v})^{d_M}) \|\mathbf{z}\|^{d_1+\cdots+d_M},$$

where $\mathbf{v} \sim \frac{\mathbf{g}}{\|\mathbf{g}\|}$. We further obtain:

$$\mathbb{E}(((\boldsymbol{\omega}^{(\mathrm{ort},1)})^\top \mathbf{z})^{d_1} \ldots ((\boldsymbol{\omega}^{(\mathrm{ort},M)})^\top \mathbf{z})^{d_M})$$

$$= \mathbb{E}((\|\boldsymbol{\omega}^{(\mathrm{ort},1)}\|)^{d_1})\ldots\mathbb{E}((\|\boldsymbol{\omega}^{(\mathrm{ort},M)}\|)^{d_M})\|\mathbf{z}\|^{d_1+\cdots+d_M}\mathbb{E}\left(\frac{\mathbf{g}_1^{d_1}\ldots\mathbf{g}_M^{d_M}}{\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{d_1+\cdots+d_M}}\right) \tag{4.23}$$

Now let us focus on the second expression from the formula on $\widehat{\Delta}(d_1,\ldots,d_M)$. We have:

$$\mathbb{E}(((\boldsymbol{\omega}^{(\mathrm{iid},1)})^\top\mathbf{z})^{d_1}\ldots((\boldsymbol{\omega}^{(\mathrm{iid},1)})^\top\mathbf{z})^{d_M}) = \prod_{m=1}^{M}\mathbb{E}(((\boldsymbol{\omega}^{(\mathrm{iid},m)})^\top\mathbf{z})^{d_m})$$

$$= \mathbb{E}((\|\boldsymbol{\omega}^{(\mathrm{iid},1)}\|)^{d_1})\ldots\mathbb{E}((\|\boldsymbol{\omega}^{(\mathrm{iid},M)}\|)^{d_M})\|\mathbf{z}\|^{d_1+\cdots+d_M}\prod_{m=1}^{M}\mathbb{E}\left(\frac{\mathbf{g}_m^{d_m}}{\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{d_m}}\right) \tag{4.24}$$

where the first equality comes from the fact that different $\boldsymbol{\omega}^{(\mathrm{iid},m)}$'s are independent and the second one is implied by the analogous analysis to the one conducted above. Next, we will need the following lemma:

Lemma 1. *For every $l\in\mathbb{N}$ such that $l\leq d$ and every $k_1,...,k_l\in\{0\}\cup\mathbb{N}$ the following holds:*

$$\mathbb{E}\left(\frac{\mathbf{g}_1^{k_1}\ldots\mathbf{g}_l^{k_l}}{\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{k_1+\cdots+k_l}}\right) = \frac{\prod_{l'=1}^{l}\mathbb{E}(\mathbf{g}_{l'}^{k_{l'}})}{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{k_1+\cdots+k_l}\right)}.$$

Proof. Take $\mathbf{r}=\frac{\mathbf{g}}{\|\mathbf{g}\|}\|\tilde{\mathbf{g}}\|$ where $\tilde{\mathbf{g}}$ is an independent copy of $\mathbf{g}$. Note that $\mathbf{r}$ has the same distribution as $\mathbf{g}$. We have:

$$\mathbb{E}(\mathbf{r}_1^{k_1})\ldots\mathbb{E}(\mathbf{r}_l^{k_l}) = \mathbb{E}(\mathbf{r}_1^{k_1}\ldots\mathbf{r}_l^{k_l}) = \mathbb{E}\left(\frac{\mathbf{g}_1^{k_1}\ldots\mathbf{g}_l^{k_l}}{\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{k_1+\cdots+k_l}}\right)\mathbb{E}(\|\tilde{\mathbf{g}}\|^{k_1+\cdots+k_l}),$$

where the first equality comes from the independence of different elements of $\mathbf{r}$ and the second equality is implied by the fact that $\tilde{\mathbf{g}}$ is independent from $\mathbf{g}$. Therefore, we have:

$$\mathbb{E}\left(\frac{\mathbf{g}_1^{k_1}\ldots\mathbf{g}_l^{k_l}}{\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{k_1+\cdots+k_l}}\right) = \frac{\mathbb{E}(\mathbf{r}_1^{k_1})\ldots\mathbb{E}(\mathbf{r}_l^{k_l})}{\mathbb{E}(\|\tilde{\mathbf{g}}\|^{k_1+\cdots+k_l})}.$$

That completes the proof since $\mathbf{r}$ and $\tilde{\mathbf{g}}$ have the same distribution as $\mathbf{g}$. □

By Lemma 1, we can rewrite the right expression from the formula on $\widehat{\Delta}(d_1,\ldots,d_M)$ as:

$$\mathbb{E}(\|\boldsymbol{\omega}^{(\mathrm{ort},1)}\|^{d_1})\ldots\mathbb{E}(\|\boldsymbol{\omega}^{(\mathrm{ort},M)}\|^{d_M})\|\mathbf{z}\|^{d_1+\cdots+d_M}\frac{\prod_{m=1}^{M}\mathbb{E}(\mathbf{g}_m^{d_m})}{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{d_1+\cdots+d_M}\right)}.$$

The left expression from the formula on $\widehat{\Delta}(d_1,\ldots,d_M)$ can be rewritten as:

$$\Lambda(d_1,\ldots,d_M)=\mathbb{E}(\|\boldsymbol{\omega}^{(\mathrm{iid},1)}\|^{d_1})\ldots\mathbb{E}(\|\boldsymbol{\omega}^{(\mathrm{iid},M)}\|^{d_M})\|\mathbf{z}\|^{d_1+\cdots+d_M}$$
$$\times\frac{\prod_{m=1}^{M}\mathbb{E}(\mathbf{g}_m^{d_m})}{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{d_1}\right)\ldots\mathbb{E}\left(\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{d_M}\right)}.$$

Clearly, $\Lambda(d_1,\ldots,d_M)$ is always nonnegative. Since marginal distributions of $\boldsymbol{\omega}^{(\mathrm{ort},m)}$ and $\boldsymbol{\omega}^{(\mathrm{iid},m)}$ are the same, we can rewrite $\widehat{\Delta}(d_1,\ldots,d_M)$ as:

$$\widehat{\Delta}(d_1,\ldots,d_M)=\Lambda(d_1,\ldots,d_M)(1-\tau(d_1,\ldots,d_M)),$$

where $\tau(d_1,\ldots,d_M)$ is defined as:

$$\tau(d_1,\ldots,d_M)=\frac{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{d_1}\right)\ldots\mathbb{E}\left(\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{d_M}\right)}{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{d_1+\cdots+d_M}\right)} \tag{4.25}$$

We need a few observations regarding $\widehat{\Delta}(d_1,\ldots,d_M)$. First, since the odd moments of the Gaussian scalar distribution $\mathcal{N}(0,1)$ are zero, $\widehat{\Delta}(d_1,\ldots,d_M)$ is zero if at least one of d_m is odd. Furthermore, $\widehat{\Delta}(d_1,\ldots,d_M)$ is trivially zero if all but at most one d_m are zero.

With our new notation, Δ can be rewritten as:

$$\Delta=\sum_{k=0}^{\infty}\frac{\phi^k}{k!}\sum_{(k_1,\ldots,k_M)\in\mathcal{S}_k}\binom{k}{k_1,\ldots,k_M}\sum_{(d_1,\ldots,d_M)\in\mathcal{D}(k_1,\ldots,k_M)}\widehat{c}_{j_1,\ldots,j_M}(d_1,\ldots,d_M)\Lambda(d_1,\ldots,d_M)$$
$$\times(1-\tau(d_1,\ldots,d_M)).$$

We also have:

$$\mathbb{E}(\exp(\phi(Z_1^{\mathrm{iid}}+\cdots+Z_M^{\mathrm{iid}})))$$
$$=\sum_{k=0}^{\infty}\frac{\phi^k}{k!}\sum_{(k_1,\ldots,k_M)\in\mathcal{S}_k}\binom{k}{k_1,\ldots,k_M}\sum_{(d_1,\ldots,d_M)\in\mathcal{D}(k_1,\ldots,k_M)}\widehat{c}_{d_1,\ldots,d_M}(k_1,\ldots,k_M)\Lambda(d_1,\ldots,d_M).$$

For the remaining case of $d_1,\dots,d_M$, i.e. when at least two d_{m_1}, d_{m_2} for $m_1 \neq m_2$ are nonzero and all d_m are even, we prove the following upper bound on τ:

Lemma 2. *The following holds if, for some $m_1 \neq m_2$, $d_{m_1}, d_{m_2} > 0$ and all d_m are even:*

$$\tau(d_1,\dots,d_M) \leq \frac{d}{d+2}.$$

Proof. Note that $\tau(d_1,\dots,d_M)$ can be rewritten as:

$$\tau(d_1,\dots,d_M) = \frac{\prod_{m=1}^{M} \mu_d(d_m)}{\mu_d(\sum_{m=1}^{M} d_m)}, \tag{4.26}$$

where $\mu_d(k)$ stands for the k'th moment of the $\chi(d)$-distribution with d degrees of freedom. Note that $\mu_d(k) = 2^{\frac{k}{2}} \frac{\Gamma(\frac{d+k}{2})}{\Gamma(\frac{d}{2})}$, where $\Gamma(\cdot)$ is the Gamma function.

Using the fact that: $\Gamma(k) = (k-1)!$ and $\Gamma(k+\frac{1}{2}) = \frac{(2k-1)!!}{2^k}\sqrt{\pi}$ for $k \in \mathbb{N}$, it is easy to see that for a fixed d, the right-hand side of the (4.26) is maximized when $d_{m_1} = d_{m_2} = 2$ and $d_{m_3} = 0$ for some $m_1 \neq m_2$ and $m_3 \notin \{m_1, m_2\}$. Furthermore, straightforward calculations show that in that case the value of the right-hand side from (4.26) is $\frac{d}{d+2}$. That completes the proof of the Lemma. □

By $\mathscr{D}'(k_1,\dots,k_M)$ denote a subset of $\mathscr{D}(k_1,\dots,k_M)$ formed by only keeping $d_1,\dots,d_M$ such that for some $m_1 \neq m_2$, $d_{m_1}, d_{m_2} > 0$ and all d_m are even. As we have shown above, $\widehat{\Delta}(d_1,\dots,d_M) = 0$ when $(d_1,\dots,d_M) \notin \mathscr{D}'(k_1,\dots,k_M)$. Otherwise,

$$\widehat{\Delta}(d_1,\dots,d_M) \geq \frac{2}{d+2}\Lambda(d_1,\dots,d_M) \geq 0.$$

Hence, since all terms in the sum

$$\Delta = \sum_{k=0}^{\infty} \frac{\phi^k}{k!} \sum_{(k_1,\dots,k_M)\in\mathscr{S}_k} \binom{k}{k_1,\dots,k_M} \sum_{(d_1,\dots,d_M)\in\mathscr{D}(k_1,\dots,k_M)} \widehat{c}_{j_1,\dots,j_M}(d_1,\dots,d_M)\widehat{\Delta}(d_1,\dots,d_M). \tag{4.27}$$

are nonnegative, we'll get a lower bound on Δ by only taking a subset of these terms. For this subset, we take $k = 4$, a subset of $\mathscr{S}_4$ with only two nonzero $k_{m_1} = k_{m_2} = 2$ for some $m_1 \neq m_2$ (there are $\binom{M}{2}$ combinations of such $k_1,\dots,k_M$). Then, we take only those $d_1,\dots,d_M$ from $\mathscr{D}(k_1,\dots,k_M)$ which correspond to $k' = 1$ in (4.21) for m_1, m_2 and $k' = 0$ for all other m's. Hence, $d_{m_1} = d_{m_2} = 2$ and all other d_m's are zero and the corresponding weight from the second sum in (4.27) would be $a_1^2 a_0^{M-2}$. For $d_1,\dots,d_M$ in such set, we'll have $\tau(d_1,\dots,d_M) \leq \frac{d}{d+2}$ by Lemma 2 and, hence, $\widehat{\Delta}(d_1,\dots,d_M) \geq \frac{2}{d+2}\Lambda(d_1,\dots,d_M)$. As

the result, we get the following lower bound on Δ:

$$\begin{aligned}\Delta &\geq \frac{2\phi^4}{4!(d+2)}\binom{M}{2}\binom{4}{2,2,0,\ldots,0}a_1^2 a_0^{M-2}\Lambda(2,2,0,\ldots,0)\\ &= \frac{\phi^4 M(M-1)}{4(d+2)}a_1^2 a_0^{M-2}\Lambda(2,2,0,\ldots,0)\\ &= \frac{\phi^4 M(M-1)}{4(d+2)}a_1^2 a_0^{M-2}\|\mathbf{z}\|^4\left(\mathbb{E}\|\boldsymbol{\omega}\|^2\right)^2\frac{(\mathbb{E}(\mathbf{g}_1^2))^2}{(\mathbb{E}\|\mathbf{g}\|^2)^2}.\end{aligned}$$

Since $\mathbf{g}\sim\mathcal{N}(0,1)^d$, $\mathbb{E}\mathbf{g}_1^2=1$ and $\mathbb{E}\|\mathbf{g}\|^2=d\mathbb{E}\mathbf{g}_1^2=d$. This results in

$$\Delta \geq \frac{\phi^4 M(M-1)}{4d^2(d+2)}a_1^2 a_0^{M-2}\|\mathbf{z}\|^4\left(\mathbb{E}\|\boldsymbol{\omega}\|^2\right)^2 \tag{4.28}$$

which concludes the proof. □

Proof of Theorem 11

Proof. We will use the notation from the proof of Theorem 10. We first consider the case $M\leq d$. We have:

$$\mathrm{Var}(\widehat{\mathscr{F}}_M^{\mathrm{iid}}(\mathbf{z})) = \mathbb{E}((\widehat{\mathscr{F}}_M^{\mathrm{iid}}(\mathbf{z}))^2) - \mathscr{F}^2_{\Omega,\mathscr{G}}(\mathbf{z}).$$

Similarly,

$$\mathrm{Var}(\widehat{\mathscr{F}}_M^{\mathrm{ort}}(\mathbf{z})) = \mathbb{E}((\widehat{\mathscr{F}}_M^{\mathrm{ort}}(\mathbf{z}))^2) - \mathscr{F}^2_{\Omega,\mathscr{G}}(\mathbf{z}).$$

We have:

$$\mathbb{E}((\widehat{\mathscr{F}}_M^{\mathrm{iid}}(\mathbf{z}))^2) = \frac{1}{M^2}\sum_{m=1}^{M}\mathbb{E}((Z_m^{\mathrm{iid}})^2) + \frac{1}{M^2}\sum_{m_1\neq m_2}\mathbb{E}(Z_{m_1}^{\mathrm{iid}}Z_{m_2}^{\mathrm{iid}}).$$

Similarly, we get:

$$\mathbb{E}((\widehat{\mathscr{F}}_M^{\mathrm{ort}}(\mathbf{z}))^2) = \frac{1}{M^2}\sum_{m=1}^{M}\mathbb{E}((Z_m^{\mathrm{ort}})^2) + \frac{1}{M^2}\sum_{m_1\neq m_2}\mathbb{E}(Z_{m_1}^{\mathrm{ort}}Z_{m_2}^{\mathrm{ort}}).$$

Therefore, since marginal distributions of Z_m^{iid} and Z_m^{ort} are the same, we have:

$$\begin{aligned}\mathrm{Var}(\widehat{\mathscr{F}}_M^{\mathrm{iid}}(\mathbf{z})) - \mathrm{Var}(\widehat{\mathscr{F}}_M^{\mathrm{ort}}(\mathbf{z})) &= 2\binom{M}{2}\frac{1}{M^2}(\mathbb{E}(Z_1^{\mathrm{iid}}Z_2^{\mathrm{iid}}) - \mathbb{E}(Z_1^{\mathrm{ort}}Z_2^{\mathrm{ort}}))\\ &= \frac{M-1}{M}(\mathbb{E}(Z_1^{\mathrm{iid}}Z_2^{\mathrm{iid}}) - \mathbb{E}(Z_1^{\mathrm{ort}}Z_2^{\mathrm{ort}}))\end{aligned} \tag{4.29}$$

Plugging in the formula for Z_m^{ort} and Z_m^{iid} from (4.20) and (4.21) and using the analysis from the proof of Theorem 10, we obtain:

$$\begin{aligned}\operatorname{Var}(\widehat{\mathscr{F}}_M^{\text{iid}}(\mathbf{z})) - \operatorname{Var}(\widehat{\mathscr{F}}_M^{\text{ort}}(\mathbf{z})) = \frac{M-1}{M} \sum_{k',k''=0}^{\infty} a_{k'} a_{k''} \|\mathbf{z}\|^{k'+k''} \mathbb{E}(\|\boldsymbol{\omega}\|^{k'}) \mathbb{E}(\|\boldsymbol{\omega}\|^{k''}) \\ \times \frac{\mathbb{E}(\mathbf{g}_1^{k'}) \mathbb{E}(\mathbf{g}_1^{k''})}{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2 + \cdots + \mathbf{g}_d^2}^{k'}\right) \mathbb{E}\left(\sqrt{\mathbf{g}_1^2 + \cdots + \mathbf{g}_d^2}^{k''}\right)} (1 - \tau(k',k'')) \end{aligned} \tag{4.30}$$

for $\boldsymbol{\omega} \sim \Omega$ and $\mathbf{g} \sim \mathscr{N}(0,1)^d$. Also, by plugging the power series expansion of $\mathscr{G}$ into the definition of $\mathscr{F}_{\Omega,\mathscr{G}}$ and using the analysis from the proof of Theorem 8, we deduce:

$$\mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z}) = \sum_{k=0}^{\infty} a_k \|\mathbf{z}\|^k \mathbb{E}(\|\mathbf{z}\|^k) \frac{\mathbb{E}(\mathbf{g}_1^k)}{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2 + \cdots + \mathbf{g}_d^2}^k\right)}. \tag{4.31}$$

Based on the definition of τ (4.25), if $k' = 0$ or $k'' = 0$, $\tau(k',k'') = 1$ and the whole corresponding term in the sum (4.30) is zero. Also, if k' is odd, $\mathbb{E}(r^{k'}) = 0$ and, again, the corresponding term in the sum (4.30) is zero. Same holds for k'' from (4.30) and k from (4.31). Hence, we can rewrite the sums above by excluding terms which are definitely zero:

$$\begin{aligned}\operatorname{Var}(\widehat{\mathscr{F}}_M^{\text{iid}}(\mathbf{z})) - \operatorname{Var}(\widehat{\mathscr{F}}_M^{\text{ort}}(\mathbf{z})) = \frac{M-1}{M} \sum_{k',k''=1}^{\infty} a_{2k'} a_{2k''} \|\mathbf{z}\|^{2k'+2k''} \mathbb{E}(\|\boldsymbol{\omega}\|^{2k'}) \mathbb{E}(\|\boldsymbol{\omega}\|^{2k''}) \\ \times \frac{\mathbb{E}(\mathbf{g}_1^{2k'}) \mathbb{E}(\mathbf{g}_1^{2k''})}{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2 + \cdots + \mathbf{g}_d^2}^{2k'}\right) \mathbb{E}\left(\sqrt{\mathbf{g}_1^2 + \cdots + \mathbf{g}_d^2}^{2k''}\right)} (1 - \tau(2k',2k'')), \end{aligned} \tag{4.32}$$

$$\mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z}) = \sum_{k=0}^{\infty} a_{2k} \|\mathbf{z}\|^{2k} \mathbb{E}(\|\mathbf{z}\|^{2k}) \frac{\mathbb{E}(\mathbf{g}_1^{2k})}{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2 + \cdots + \mathbf{g}_d^2}^{2k}\right)}.$$

According to Lemma 2, for all k',k'' in the sum (4.32), we have $\tau(k',k'') \leq \frac{d}{d+2}$. Hence, we get:

$$\begin{aligned}\operatorname{Var}(\widehat{\mathscr{F}}_M^{\text{iid}}(\mathbf{z})) - \operatorname{Var}(\widehat{\mathscr{F}}_M^{\text{ort}}(\mathbf{z})) \geq \frac{2(M-1)}{M(d+2)} \sum_{k',k''=0}^{\infty} a_{2k'} a_{2k''} \|\mathbf{z}\|^{2k'+2k''} \mathbb{E}(\|\boldsymbol{\omega}\|^{2k'}) \mathbb{E}(\|\boldsymbol{\omega}\|^{2k''}) \\ \times \frac{\mathbb{E}(\mathbf{g}_1^{2k'}) \mathbb{E}(\mathbf{g}_1^{2k''})}{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2 + \cdots + \mathbf{g}_d^2}^{2k'}\right) \mathbb{E}\left(\sqrt{\mathbf{g}_1^2 + \cdots + \mathbf{g}_d^2}^{2k''}\right)} \end{aligned}$$

$$= \frac{2(M-1)}{M(d+2)} \left(\sum_{k=1}^{\infty} a_{2k} \|\mathbf{z}\|^{2k} \mathbb{E}(\|\boldsymbol{\omega}\|^{2k}) \frac{\mathbb{E}(\mathbf{g}_1^{2k})}{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2 + \cdots + \mathbf{g}_d^2}^{2k}\right)} \right)^2$$

$$= \frac{2(M-1)}{M(d+2)} (\mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z}) - a_0)^2. \tag{4.33}$$

According to (4.31), $a_0 = \mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{0}_d)$ which completes the proof for the case $M \leq d$.

When $M > d$, M/d is integer, observe that random ordered sets $\{(\boldsymbol{\omega}^{(\mathrm{ort},1+m'd)}, \ldots, \boldsymbol{\omega}^{(\mathrm{ort},d+m'd)})\}_{m'=0}^{M/d-1}$ are independent, so we can write:

$$\mathrm{Var}\widehat{\mathscr{F}}_M^{\mathrm{ort}}(\mathbf{z}) = \frac{1}{M^2} \sum_{m'=0}^{M/d-1} \mathrm{Var}\left(\sum_{l=0}^{d} \mathscr{G}((\boldsymbol{\omega}^{(\mathrm{ort},l+m'd)})^\top \mathbf{z}) \right)$$

$$= \frac{d^2}{M^2} \sum_{m'=0}^{M/d-1} \mathrm{Var}\left(\frac{1}{d} \sum_{l=0}^{d} \mathscr{G}((\boldsymbol{\omega}^{(\mathrm{ort},l+m'd)})^\top \mathbf{z}) \right). \tag{4.34}$$

We apply (4.33) to each variance in the sum above which can be thought as an orthogonal estimator of $\mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z})$ with d Monte-Carlo samples:

$$\mathrm{Var}\left(\frac{1}{d} \sum_{l=0}^{d} \mathscr{G}((\boldsymbol{\omega}^{(\mathrm{ort},l+m'd)})^\top \mathbf{z}) \right) \leq \mathrm{Var}\left(\frac{1}{d} \sum_{l=0}^{d} \mathscr{G}((\boldsymbol{\omega}^{(\mathrm{iid},l+m'd)})^\top \mathbf{z}) \right)$$
$$- \frac{2(d-1)}{d(d+2)} (\mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z}) - \mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{0}_d))^2$$

Now we plug this into (4.34):

$$\widehat{\mathscr{F}}_M^{\mathrm{iid}}(\mathbf{z}) \leq \frac{d^2}{M^2} \left(\sum_{m'=0}^{M/d-1} \mathrm{Var}\left(\frac{1}{d} \sum_{l=0}^{d} \mathscr{G}((\boldsymbol{\omega}^{(\mathrm{iid},l+m'd)})^\top \mathbf{z}) \right) \right.$$
$$\left. - \frac{2(d-1)}{d(d+2)} (\mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z}) - \mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{0}_d))^2 \right)$$
$$= \frac{1}{M^2} \sum_{m'=0}^{M/d-1} \mathrm{Var}\left(\sum_{l=0}^{d} \mathscr{G}((\boldsymbol{\omega}^{(\mathrm{iid},l+m'd)})^\top \mathbf{z}) \right) - \frac{2(d-1)}{M(d+2)} (\mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z}) - \mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{0}_d))^2$$
$$= \widehat{\mathscr{F}}_M^{\mathrm{iid}}(\mathbf{z}) - \frac{2(d-1)}{M(d+2)} (\mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{z}) - \mathscr{F}_{\Omega,\mathscr{G}}(\mathbf{0}_d))^2.$$

That completes the proof of the case $M > d$, integer M/d. □

Chapter 5

SLiM Performer: beyond linear memory consumption

5.1 Motivation

As demonstrated in Chapter 3 (Section 3.4), the computational and memory complexity of the self-attention block in Performers is $O(L)$ as a function of the sequence length L. Since there is a fixed number of self-attention blocks, and the computational and memory complexity of the feedforward block is also $O(L)$, we conclude that the total computational and memory complexity of Performer is $O(L)$. In this Chapter, we discover yet another remarkable property of Performers: in the case of causal Performers, the memory complexity for the forward and backward pass can be decreased up to $O(1)$ while the computational complexity remains $O(L)$. Notably, no approximations are introduced, so the obtained gradient is correct and backward-compatible.

If, during training via the minibatch gradient descent, the batch size is bigger than one, a *gradient accumulation* technique (Ott et al., 2018) can be used to reduce memory complexity while retaining the same amount of computations. The idea is simple: instead of evaluating the whole batch at once, evaluate instances in the batch one by one and accumulate the resulting gradient. This way, a practitioner only needs the amount of memory required to process a single instance. Unfortunately, if the batch size is one, gradient accumulation cannot be used and some other way to improve memory efficiency is required.

This challenge is especially pronounced for deeper Performer models and longer sequences when the batch size becomes smaller to fit the model into memory and might as well hit the smallest possible value of one. Also, the scenario of using the batch size of one during gradient descent can be imagined when using Performers on the client's device

(e.g. smartphones, embedded devices or microcontrollers) and fine-tuning it on client's data (personalization of the model weights).

Hence, the memory-efficient forward- and backward-propagation algorithm, which we refer to as *Sub-Linear Performer* (or *SLiM Performer*), is motivated by important practical applications in decentralized and democratized deep learning. The chapter has the following structure:

- Section 5.2 establishes a compact notation for Performer which is convenient for subsequent derivations.
- Section 5.3 describes an algorithm for the forward pass through the whole Performer with $O(L)$ computational complexity, $O((L/E)\log E)$ parallel time complexity and $O(E)$ memory plus a negligible storage for the input sequence where E is a user-defined integer constant, e.g. 1.
- Section 5.4 extends the forward pass algorithm to the computation of the gradient of the loss with respect to Performer's parameters $\boldsymbol{\theta}$. The computational complexity, parallel time complexity and memory complexity are the same as for the forward pass.
- Section 5.5 presents the non-asymptotic analysis of the amount of floating-point operations (FLOPs) of the SLiM-Performer algorithm. We establish that memory efficiency of gradient computation via SLiM-Performer is FLOPs-equivalent to two forward and one backward pass through the vanilla $O(L)$ Performer which is a small price to pay. Also, the section performs a conceptual comparison of SLiM-Performer with other deep learning architectures for sequence processing.
- Section 5.6 is experimental evaluation of SLiM-Performer with time and memory benchmarking and downstream performance comparison.
- Finally, Section 5.7 is reserved for discussions.

5.2 Compact notation for Performer

We consider a causal Performer, that is the parameterized architecture $\mathbf{X}^{(0)} \to \mathbf{X}^{(1)} \to \cdots \to \mathbf{X}^{(s)} \to \mathbf{X}^{(\text{out})}$ as in Section 2.2 but with causal generalized self-attention $\text{Att}^{\text{GA}}_{\to}$ (Section 3.6) instead of $\text{Att}_{\square}$ in (2.15). Since FAVOR+ is a special case of GA (Section 4.5.1), we don't lose generality by considering Performers with GA.

As we discussed in Section 3.3 and Section 3.6, all causal efficient attention reduces to computing prefix sums $\mathbf{R} = \text{PS}((\mathbf{C}_i \times (\mathbf{P}_i^{(2)})^\top)_{i=1}^L)$, where $\mathbf{P}_i^{(2)} = g(\mathbf{K}_i)$ and $\mathbf{C} = \begin{bmatrix}\mathbf{V} & \mathbf{1}_L\end{bmatrix}$,

along the sequence dimension. It can be observed that *all* information propagation between elements of the sequence in causal Performers happens inside prefix sum. More formally, we can rewrite transformations $\mathbf{X}^{(0)} \to \mathbf{X}^{(1)} \to \cdots \to \mathbf{X}^{(s)}$ as follows. For each $1 \leq r \leq s$,

$$\mathbf{T}^{(r-1)}, \mathbf{\Gamma}^{(r-1)} = F^{(r)}(\mathbf{X}^{(r-1)}; \boldsymbol{\theta}), \tag{5.1}$$

$$\mathbf{U}^{(r-1)} = \mathrm{PS}(\mathbf{T}^{(r-1)}), \tag{5.2}$$

$$\mathbf{X}^{(r)} = G^{(r)}(\mathbf{U}^{(r-1)}, \mathbf{\Gamma}^{(r-1)}; \boldsymbol{\theta}). \tag{5.3}$$

Here $\boldsymbol{\theta} \in \mathbb{R}^{n_{param}}$ is a set of all parameters, $\mathbf{T}^{(r-1)}, \mathbf{U}^{(r-1)} \in \mathbf{R}^{L \times D_1}$ and $\mathbf{\Gamma}^{(r-1)} \in \mathbf{R}^{L \times D_2}$ are the following matrices (see Figure 5.1 for an illustration).

- $\mathbf{T}^{(r-1)}$ is a matrix of representations which are passed into the prefix sum operator $\mathrm{PS}(\cdot)$. That is, for each $1 \leq i \leq L$, $\mathbf{T}_i^{(r-1)}$ is a concatenation of flattened $\mathbf{C}_i \times g(\mathbf{K}_i)^\top$ for all h self-attention heads computed at the r'th layer. Consequently, $D_1 = M(d+1)h$.
- For each $1 \leq i \leq L$, $\mathbf{U}_i^{(r-1)}$ is a concatentation of flattened $\mathbf{R}_l$ – results of the prefix sum inside each self-attention head at r'th layer.
- $\mathbf{\Gamma}^{(r-1)}$ is a matrix of representations which skip the prefix sum. For each $1 \leq i \leq L$, $\mathbf{\Gamma}_i^{(r-1)}$ is a concatenation of $\mathbf{X}_i^{(r-1)}$ and the series of $g(\mathbf{Q}_i)$ for each out of h attention heads (2.15). Therefore, $D_2 = Mh + d_{\text{hid}}$.

$F^{(r)}$ and $G^{(r)}$ are mappings when depend on model's parameters $\boldsymbol{\theta}$. That is, they take subsets of $\boldsymbol{\theta}$ corresponding to r'th layer weights. $F^{(r)}$ is responsible for constructing $\mathbf{T}^{(r-1)}$ and $\mathbf{\Gamma}^{(r-1)}$ – representations which precede prefix sum, while $G^{(r)}$ finalizes multi-head self-attention and includes the feed-forward block $\mathrm{FFN}(\cdot)$ (2.14). Importantly, $F^{(r)}$ and $G^{(r)}$ are applied rowwise, i.e. (5.1-5.3) can be rewritten as

$$\forall 1 \leq i \leq L : \mathbf{T}_i^{(r-1)}, \mathbf{\Gamma}_i^{(r-1)} = F^{(r)}(\mathbf{X}_i^{(r-1)}; \boldsymbol{\theta}), \quad \mathbf{X}_i^{(r)} = G^{(r)}(\mathbf{U}_i^{(r-1)}, \mathbf{\Gamma}_i^{(r-1)}; \boldsymbol{\theta}).$$

Hence indeed, the only place where the signal is propagated across the sequence is the prefix sum in (5.1-5.3).

The representation (5.1-5.3) encapsulates architecture details of Performer inside $\{F^{(1)}, G^{(1)}, \ldots, F^{(s)}, G^{(s)}\}$. In fact, the representation (5.1-5.3) holds for various possible modifications proposed in the literature. This includes but is not limited by the different positioning of layer normalization (Vaswani et al., 2017; Xiong et al., 2020), addition of a stabilizing gating mechanism (Parisotto et al., 2020), weight sharing across layers (Lan et al., 2020) or reversible Transformer layers (Kitaev et al., 2020).

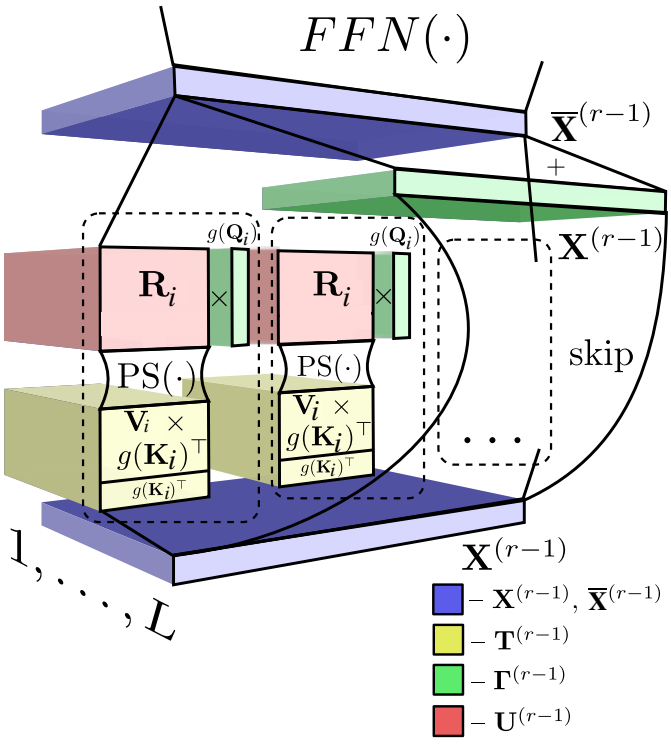


Fig. 5.1 r'th layer and its decomposition into $\mathbf{T}^{(r-1)}, \mathbf{\Gamma}^{(r-1)}, \mathbf{U}^{(r-1)}$.

5.3 Memory-efficient forward pass through Performer

Suppose the memory budget is not enough to perform a complete forward pass through Performer (5.1-5.3 for $r = 1, \ldots, s$), because the input sequence length L is too big. We show that instead we can emulate the full forward computation under the memory needed for a forward pass through the input of length $E \leq L$ plus a small addition. $1 \leq E \leq L$ is arbitrary and user-defined.

Split matrices $\mathbf{X}^{(r)}, \mathbf{T}^{(r)}, \mathbf{\Gamma}^{(r)}, \mathbf{U}^{(r)}$, into $P = \lceil L/E \rceil$ slices of size at most E along the vertical axis: for each $1 \leq p \leq P$,

$$\mathbf{X}^{(r,p)} = (\mathbf{X}^{(r)}_{A_p+i})_{i=1}^{B_p} \in \mathbb{R}^{B_p \times d_{\text{hid}}}, \quad \mathbf{T}^{(r,p)} = (\mathbf{T}^{(r)}_{A_p+i})_{i=1}^{B_p},$$
$$\mathbf{U}^{(r,p)} = (\mathbf{U}^{(r)}_{A_p+i})_{i=1}^{B_p} \in \mathbb{R}^{B_p \times D_1}, \quad \mathbf{\Gamma}^{(r,p)} = (\mathbf{\Gamma}^{(r)}_{A_p+i})_{i=1}^{B_p} \in \mathbb{R}^{B_p \times D_2},$$

where $A_p = (p-1)E$ and by B_p, $1 \leq p \leq P$, we denote size of the p'th slice: $B_p = E$ for $p < P$, $B_P \leq E$. Based on (5.1-5.3), we conclude that for each $1 \leq p \leq P$ and $1 \leq r \leq s$ it holds that

$$\mathbf{T}^{(r-1,p)}, \mathbf{\Gamma}^{(r-1,p)} = F^{(r)}(\mathbf{X}^{(r-1,p)}; \boldsymbol{\theta}), \tag{5.4}$$

$$\mathbf{U}^{(r-1,p)} = \mathbf{1}_{B_p} \times (\mathbf{U}_{B_{p-1}}^{(r-1,p-1)})^\top + \mathrm{PS}(\mathbf{T}^{(r-1,p)}), \tag{5.5}$$

$$\mathbf{X}^{(r,p)} = G^{(r)}(\mathbf{U}^{(r-1,p)}, \mathbf{\Gamma}^{(r-1,p)}; \boldsymbol{\theta}). \tag{5.6}$$

Here, we denote $\mathbf{U}_{B_0}^{(r-1,0)} = \mathbf{0}_{D_1}$.

Now, instead of iterating over $r = 1, \ldots s$ and computing (5.1-5.3) for the whole sequence at once, we first iterate over $p = 1, \ldots, P$ and then iterate over $r = 1, \ldots, s$ in a nested loop to compute (5.4-5.6). As can be deduced from the (5.4-5.6), we only need to maintain the current value of $(\mathbf{U}_{B_{p-1}}^{(r-1,p-1)})_{r=1}^{s} \in \mathbb{R}^{s \times D_1}$ in the outer iteration over p.

We further assume that we use causal Performer for language modelling with an input string $\mathbf{p} \in \Sigma^L$ as in Section 2.2.3, and our goal is to evaluate the negative log-likelihood loss $\mathscr{L}$ (2.20) (the algorithm for sampling has a similar form).

Denote $\mathbf{\Pi}^{(p)} = (\mathbf{U}_{B_p}^{(r-1,p)})_{r=1}^{s} \in \mathbb{R}^{s \times D_1}$, $0 \leq p \leq P$. The memory-efficient algorithm for the forward pass is as follows. First, initialize $\mathscr{L} = 0$ and $\mathbf{\Pi}^{(0)} = \mathbf{0}_{s \times D_1}$. Then, iterate over $p = 1, \ldots, P$ and maintain the current value of $\mathbf{\Pi}^{(p-1)}$. During each iteration, compute $\mathbf{X}^{(0,p)} = (\mathbf{emb}_{\mathbf{p}_{A_p+i}} + \mathbf{posemb}_{A_p+i})_{i=1}^{B_p}$ as in (2.19). Then iterate over $r = 1, \ldots, s$ where compute (5.4-5.6) and update $\mathbf{\Pi}_r^{(p)} = \mathbf{U}_{B_p}^{(r-1,p)}$. Finally, compute $\mathbf{X}^{(out,p)} = (\mathbf{W}^{(\mathrm{out})}\mathbf{X}_i^{(s,p)} + \mathbf{b}^{(\mathrm{out})})_{i=1}^{B_p}$ and update $\mathscr{L} \mathrel{+}= \mathscr{L}^{(p)}(\mathbf{X}^{(out,p)})$, where we denote:

$$\mathscr{L}^{(p)}(\mathbf{X}^{(out,p)}) = -L^{-1} \sum_{i=1}^{B_p} \log \left(\frac{\exp\left(\mathbf{X}_{i,\mathbf{p}_i}^{(\mathrm{out},p)}\right)}{\sum_{l=1}^{d_{\mathrm{out}}} \exp(\mathbf{X}_{i,l}^{(\mathrm{out},p)})} \right)$$

as in (2.18,2.20).

By the end of the iteration over p, the correct loss value (2.20) is computed. As a result, the forward pass requires $O(L)$ total computations since there are $P = O(L/E)$ iterations over p and $O(E)$ computations inside each iteration. Since inside each iteration over p we compute prefix sums of size $O(E)$ which can be parallelized into $O(\log E)$ parallel time (Section 3.4), this results in $O((L/E)\log E)$ total parallel time. Since inside every iteration over p we process chunks of size $O(E)$, the total memory consumption of the algorithm is only $O(E)$. This memory is in addition to the input sequence $\mathbf{p} \in \Sigma^L$ storage which is $O(L)$ in principle, however the constant is negligibly small. For instance, if $\mathbf{p}$ is a flattened image or an ASCII text string, then it occupies precisely L bytes in memory.

5.4 Memory-efficient backward pass through Performer

The goal of a backward pass is to compute the gradient $\nabla_{\boldsymbol{\theta}}\mathscr{L}$ of the loss function with respect to parameters $\boldsymbol{\theta}$. One can just perform automatic differentiation (Griewank and Walther, 2008) through the computation graph induced by the memory-efficient forward pass algorithm from Section 5.3. However, such a backward pass would need to store all intermediate tensors produced during the forward pass, resulting in $O(L)$ memory complexity as a function of L and E. Instead, we propose a back-propagation algorithm which has the same time and memory complexity as the efficient forward pass.

Let $\boldsymbol{\theta}^{(1)} = \cdots = \boldsymbol{\theta}^{(P)} = \boldsymbol{\theta}$ be results of a symbolic "copy operation" performed on $\boldsymbol{\theta}$, so that for all $1 \leq p \leq P$, $\boldsymbol{\theta}^{(p)}$ is used instead of $\boldsymbol{\theta}$ in (5.4-5.6). Then the total gradient of $\boldsymbol{\theta}$ has the form $\nabla_{\boldsymbol{\theta}}\mathscr{L} = \nabla_{\boldsymbol{\theta}^{(1)}}\mathscr{L} + \cdots + \nabla_{\boldsymbol{\theta}^{(P)}}\mathscr{L}$. Denote $g\boldsymbol{\Pi}^{(p)} = \nabla_{\boldsymbol{\Pi}^{(p)}}\mathscr{L}$. In Appendix 5.A, we prove the proposition:

Proposition 1. *For $1 \leq p \leq P$, let $\Phi^{(p)} : \mathbb{R}^{d_{\text{param}}} \times \mathbb{R}^{s\times D_1} \times \mathbb{R}^{s\times D_1} \to \mathbb{R}$ be defined as*

$$\Phi^{(p)}(\boldsymbol{\theta}^{(p)}, \boldsymbol{\Pi}^{(p-1)}, \mathbf{Z}) = \mathscr{L}^{(p)}(\mathbf{X}^{(out,p)}) + \sum_{r=1}^{s} \mathbf{Z}_r^\top \boldsymbol{\Pi}_r^{(p)}$$

where $\mathbf{X}^{(out,p)}$ and $\boldsymbol{\Pi}^{(p)} = (\mathbf{U}_{B_p}^{(r-1,p)})_{r=1}^{s}$ are results of (5.4-5.6) iteration over $r = 1, \ldots, s$ with parameters $\boldsymbol{\theta} = \boldsymbol{\theta}^{(p)}$ and $(\mathbf{U}_{B_{n-1}}^{(r-1,p-1)})_{r=1}^{s}$ equal to $\Phi^{(p)}$'s second argument $\boldsymbol{\Pi}^{(p-1)}$. Then, the following holds:

$$\nabla_{\boldsymbol{\theta}^{(p)}}\mathscr{L} = \nabla_{\boldsymbol{\theta}^{(p)}}\Phi^{(p)}(\boldsymbol{\theta}^{(p)}, \boldsymbol{\Pi}^{(p-1)}, g\boldsymbol{\Pi}^{(p)}), \tag{5.7}$$

$$g\boldsymbol{\Pi}^{(p-1)} = \nabla_{\boldsymbol{\Pi}^{(p-1)}}\Phi^{(p)}(\boldsymbol{\theta}^{(p)}, \boldsymbol{\Pi}^{(p-1)}, g\boldsymbol{\Pi}^{(p)}). \tag{5.8}$$

Further, we have:

$$g\boldsymbol{\Pi}^{(P)} = \mathbf{0}_{r\times D_1}.$$

Gradients $\nabla_{\boldsymbol{\theta}^{(p)}}\Phi^{(p)}$ and $g\boldsymbol{\Pi}^{(p-1)}$ in (5.7,5.8) can be computed by a single automatic differentiation through $\Phi^{(p)}$.

An efficient way to compute and sum all $\nabla_{\boldsymbol{\theta}^{(p)}}\mathscr{L}$ is to iterate in a backward direction $p = P, \ldots, 1$ and to maintain values of $\boldsymbol{\Pi}^{(p)}, g\boldsymbol{\Pi}^{(p)}$. $\boldsymbol{\Pi}^{(P)}$ is known after the end of the forward pass, and for all $1 \leq p \leq P$,

$$\forall 1 \leq p \leq P: \quad \boldsymbol{\Pi}^{(p-1)} = \boldsymbol{\Pi}^{(p)} - \sum_{i=1}^{B_p} (\mathbf{T}_i^{(r-1,p)})_{r=1}^{s}. \tag{5.9}$$

Further, according to Proposition 1, $\boldsymbol{g\Pi}^{(P)} = \mathbf{0}_{r\times D_1}$ and for $1 \leq p \leq P$, $\boldsymbol{g\Pi}^{(p-1)}$ can be updated from $\boldsymbol{g\Pi}^{(p)}$ via (5.8).

If $\mathbf{w}$ is some vector of length B_p and $\mathscr{H}$ is some scalar function of $\mathbf{v} = \mathrm{PS}(\mathbf{w})$, then for all $1 \leq i \leq B_p$: $\nabla_{\mathbf{w}_i}\mathscr{H} = \sum_{j=i}^{B_p} \nabla_{\mathbf{v}_j}\mathscr{H}$. In other words, the gradient through $\mathrm{PS}(\cdot)$ is another prefix sum computed backwards. Hence, automatic differentiation through $\Phi^{(p)}$ takes the same parallel time $O(\log E)$, serial time $O(E)$ and memory $O(E)$, as the forward computation of $\Phi^{(p)}$. Since during the whole backward pass algorithm, we only store and update tensors $\boldsymbol{\Pi}^{(p)}, \boldsymbol{g\Pi}^{(p)}$ the size of which doesn't depend on L and E, this results in total $O((L/E)\log E)$ parallel time, $O(L)$ serial time and $O(E)$ memory in addition to $\mathbf{p}$ storage. A full description of the forward-backward pass is presented in Algorithm 5. Figure 5.2 is an illustration of the algorithm. We refer to Algorithm 5 as *sub-linear memory Performer (SLiM Performer)*.

Algorithm 5 Low-memory forward-backward pass. See Algorithm 6 for updateProc. Compared to the notation from text, redundant indices are dropped and tensor names are reused here and in Algorithm 6.

Input: $\mathbf{p} \in \Sigma^L$, $\boldsymbol{\theta} \in \mathbb{R}^{n_{\text{param}}}$, $E \in \mathbb{N}$.
Output: loss $\mathscr{L}$, gradient $\nabla_{\boldsymbol{\theta}}\mathscr{L}$.
Set $\mathscr{L} := 0$, $\boldsymbol{\Pi} := \mathbf{0}_{s\times D_1}$;
for $p = 1$ **to** P **do**
    updateProc(p, False);
end for
Set $\nabla_{\boldsymbol{\theta}}\mathscr{L} := \mathbf{0}_{n_{\text{param}}}$, $\boldsymbol{g\Pi} := \mathbf{0}_{s\times D_1}$;
for $p = P$ **to** 1 **do**
    updateProc(p, True);
end for
Return $\mathscr{L}$, $\nabla_{\boldsymbol{\theta}}\mathscr{L}$.

Algorithm 6 updateProc procedure.

Input: $p \in \mathbb{N}$, binary flag onBackProp .
if onBackProp **then**
  Initialize $\Phi := 0$;
end if
$\mathbf{X}^{(0)} := (\mathbf{emb}_{\mathbf{p}_{A_p+i}} + \mathbf{posemb}_{A_p+i})_{i=1}^{B_p}$;
for $r = 1$ **to** s **do**
  Compute $\mathbf{T}, \mathbf{\Gamma} := F^{(r)}(\mathbf{X}; \boldsymbol{\theta})$;
  if onBackProp **then**
    Update $\mathbf{\Pi}_r -= \sum_{i=1}^{B_p} \mathbf{T}_i$;
  end if
  Set $\mathbf{U} := \mathbf{1}_{B_p} \mathbf{\Pi}_r^\top + \mathrm{PS}(\mathbf{T})$, $\mathbf{X} := G^{(r)}(\mathbf{U}, \mathbf{\Gamma}; \boldsymbol{\theta})$;
  if onBackProp **then**
    Update $\Phi += \boldsymbol{g}\mathbf{\Pi}_r^\top \mathbf{U}_{B_p}$;
  else
    Update $\mathbf{\Pi}_r := \mathbf{U}_{B_p}$;
  end if
end for
Set $\mathscr{L}^{(\mathrm{upd})} := \mathscr{L}^{(p)}(\mathbf{X}\mathbf{W}^{(\mathrm{out})} + \mathbf{b}^{(\mathrm{out})})$;
if onBackProp **then**
  Update $\Phi += \mathscr{L}^{(\mathrm{upd})}$;
  Compute $\nabla_{\boldsymbol{\theta}}\Phi, \nabla_{\mathbf{\Pi}}\Phi$ through automatic differentiation;
  Update $\nabla_{\boldsymbol{\theta}}\mathscr{L} += \nabla_{\boldsymbol{\theta}}\Phi, \quad \boldsymbol{g}\mathbf{\Pi} := \nabla_{\mathbf{\Pi}}\Phi$;
else
  Set $\mathscr{L} += \mathscr{L}^{(\mathrm{upd})}$;
end if

5.5 Complexity analysis

As we have shown, Performer can be trained in parallel time $O((L/E)\log E)$ and $O(E)$ memory in addition to the input $\mathbf{p}$ storage. Hence, E is a tradeoff parameter: when E is maximal ($E = L$), the model is fully-parallelized, therefore resulting in the fastest execution. Whereas the minimum $E = 1$ corresponds to a step-by-step processing, i.e. a fully-sequential regime which doesn't benefit from parallelized computations on GPU or TPU, but consumes $O(1)$ memory as a function of L.

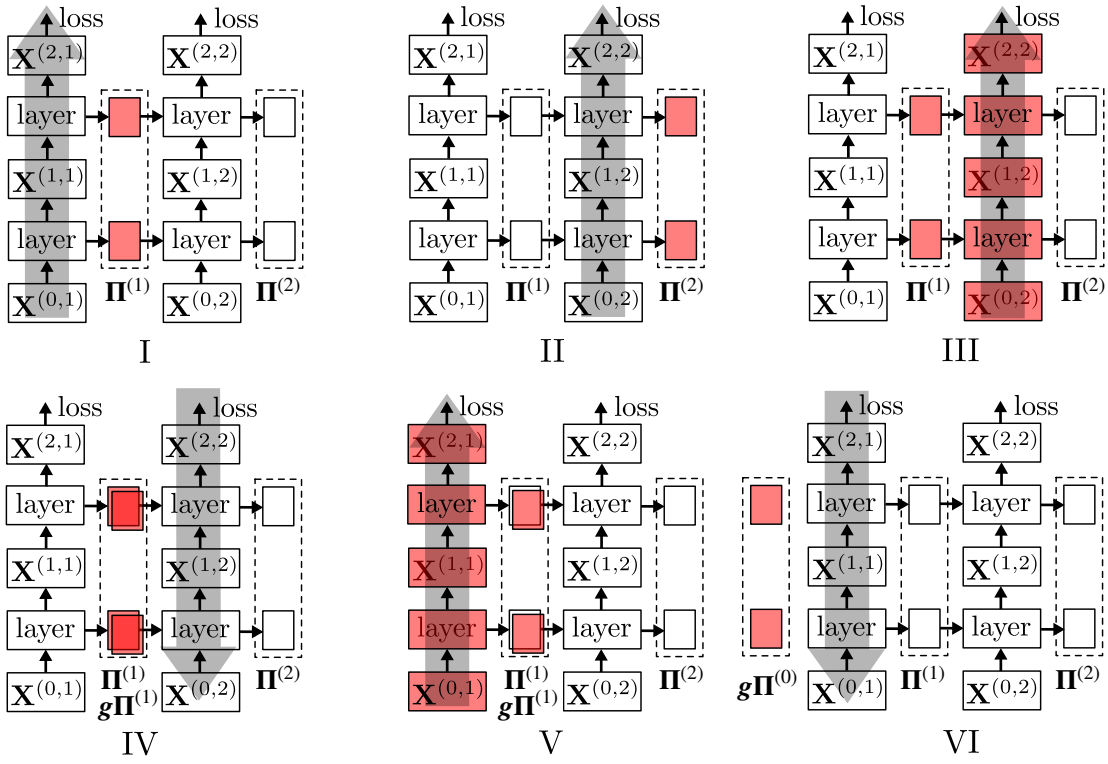


Fig. 5.2 Illustration of Algorithm 5 when $s = P = 2$. Red color indicates objects stored in memory. I-II) forward passes for $p = 1, 2$ respectively, only the loss value and $\mathbf{\Pi}^{(p)}$ are stored. III) backward pass start, forward computation through the slice $p = 2$ to build a symbolic $\Phi^{(2)}$ and update $\mathbf{\Pi}^{(2)} \to \mathbf{\Pi}^{(1)}$. IV) back-propagation through $\Phi^{(2)}$ to find $\nabla_{\boldsymbol{\theta}^{(2)}}\mathscr{L}$ and $\boldsymbol{g}\mathbf{\Pi}^{(1)}$. V,VI) the same backward iteration for $p = 1$.

During the forward pass, Algorithm 5 requires as many total FLOPs as the naive forward pass through (5.4-5.6) since the computations are simply "regrouped" into iterating over $1 \le p \le P$ first and then over $1 \le r \le s$. As for the backward pass, for each $1 \le p \le P$, the forward pass through p's slice is repeated for symbolic construction of $\Phi^{(p)}$ (see Algorithm 6), and then automatic differentiation is run through $\Phi^{(p)}$. In addition, a backward update of $\mathbf{\Pi}^{(p)}$ (5.9) is computed, taking precisely $B_p sM(d+1)h$ "add" operations. Hence, we conclude that Algorithm 5 requires as many FLOPs as two forward and one backward pass through (5.4-5.6) for the whole sequence $\mathbf{p}$ plus

$$\sum_{p=1}^{P} B_p sM(d+1)h = LsM(d+1)h = LsMd_{\text{hid}} + LsMh$$

FLOPs. To characterize this addition, assuming that typically d_{ff} (dimension of the feed-forward block) is $4d_{\text{hid}}$ in practice, observe that the application of dense Performer layers

(2.14) alone requires $3Lsd_{\text{hid}}^2 + 2Lsd_{\text{hid}}d_{\text{ff}} = 11Lsd_{\text{hid}}^2$ FLOPs. This is much bigger than $LsMd_{\text{hid}} + LsMh$, since M is smaller than d_{hid} in practice, e.g. $M = 256$, $d_{\text{hid}} = 512$ for "Regular" configuration from Sections 3.8, 4.6.

Since the back-propagation takes roughly 5 times more FLOPs than the forward pass (Griewank and Walther, 2008), we conclude that memory efficiency of Algorithm 5 results in a small constant-time increase in FLOPs. FLOPs affect energy consumption (Wu* et al., 2020), a crucial factor for on-device applications.

A further analysis of Algorithm 5 reveals that the $E = 1$ regime requires as much memory as if Transformer was applied to a sequence of length 1 plus exactly $2sd_{\text{hid}}(M+1)$ floating-point numbers for storing $\boldsymbol{\Pi}, \boldsymbol{g\Pi}$. For comparison, the subset of $\boldsymbol{\theta}$ corresponding to dense layers in self-attention and feed-forward blocks (2.14) occupies $3sd_{\text{hid}}^2 + 2sd_{\text{hid}}d_{\text{ff}} = 11sd_{\text{hid}}^2$ floating-point numbers. Again, this is much bigger than $2sd_{\text{hid}}(M+1)$, since M is smaller than d_{hid} in practice.

To understand these fruitful properties, we conceptually compare Performer with recurrent neural networks (RNNs) (Cho et al., 2014; Hochreiter and Schmidhuber, 1997) and residual networks (e.g. neural ordinary differential equations (Chen et al., 2018)) which are also used for sequence processing. The r'th layer for all models has the following form, $1 \leq i \leq L$:

$$\text{RNN}: \quad \mathbf{X}_i^{(r)} = \mathscr{T}^{(r)}(\mathbf{X}_{i-1}^{(r)}, \mathbf{X}_i^{(r-1)}), \tag{5.10}$$

$$\text{Residual}: \quad \mathbf{X}_i^{(r)} = \mathbf{X}_{i-1}^{(r)} + \mathscr{T}^{(r)}(\mathbf{X}_{i-1}^{(r)}, \mathbf{X}_i^{(r-1)}), \tag{5.11}$$

$$\text{Performer}: \quad \mathbf{X}_i^{(r)} = \mathbf{X}_{i-1}^{(r)} + \mathscr{T}^{(r)}(\mathbf{X}_i^{(r-1)}). \tag{5.12}$$

Here $\mathscr{T}^{(r)}$ is some nonlinear map. Observe that Performer is the only architecture where $\mathbf{X}_i^{(r)}$ depends linearly on $\mathbf{X}_{i-1}^{(r)}$. It's not hard to see that Algorithm 5 can be applied to any architecture of type (5.10-5.12). Despite the update's simplicity, Performer appears to work very well in challenging real-life setups, and, as shown in Chapters 3, 4, can approximate any conventional Transformer with exponential self-attention. See Table 5.1 for a complexity comparison of all discussed architectures and the proposed algorithm.

Table 5.1 Complexity for the back-propagation as a function of sequence length L and the tradeoff parameter $E \leq L$. The indicated memory complexity is in addition to the input sequence $\mathbf{p}$ storage. The serial time complexity for Performer is reported for the version with the iterative $\mathrm{PS}(\cdot)$ computation (onPS = False in Algorithm 3) while the parallel time is reported for the parallel prefix sum (onPS = True in Algorithm 3). For both methods, memory complexity is the same, but the constant is smaller in the iterative version.

Model	Serial time	Parallel time	Memory
RNN	$O(L)$	$O(L)$	$O(L)$
Residual NN	$O(L)$	$O(L)$	$O(L)$
Transformer	$O(L^2)$	$O(\log L)$	$O(L^2)$
Performer	$O(L)$	$O(\log L)$	$O(L)$
SLiM Performer	$O(L)$	$O((L/E)\log E)$	$O(E)$
SLiM Performer $E = 1$	$O(L)$	$O(L)$	$\boldsymbol{O(1)}$

5.6 Experiments

In the experimental section, we aim to answer the following questions about using SLiM-Performer algorithm in practice:

1. Does the theoretical time-memory tradeoff, controlled by E, agree with benchmarks of time and memory for varied E?

2. In precise arithmetic, different values of E lead to the same correct gradient $\nabla_{\boldsymbol{\theta}}\mathscr{L}$. Does this hold in practice, when finite-precision arithmetic is employed?

3. Can a model, pre-trained with a bigger E (e.g. on a server), be fine-tuned with a smaller E (e.g. on an embedded device)? Does E affect the result of training from scratch?

We address these questions in subsections below. We analyse 4 model configurations (L, s, d_{hid}): I = (8192, 1, 1024), II = (1024, 3, 512), III = (4096, 3, 1024), IV = (16384, 3, 1024). In all configurations, we set $d_{\text{ff}} = 4d_{\text{hid}}$, $d = 64$ (a standard choice for the query dimension in Transformers) and, consequently, $h = d_{\text{hid}}/64$ (number of heads). We set $M = d$ and employ $g(\mathbf{x}) = (\mathbf{x}_i^2)_{i=1}^d$, i.e. we use Performer-SQR with identity as $\boldsymbol{\Omega}$ (Section 3.6) which we find to work well. In all experiments, $\Sigma = \{0, \ldots, 255\}$ and batch size is set to 1, i.e. we analyse a setup where gradient accumulation cannot be used to decrease memory, and therefore our algorithm is crucial. We use a single NVIDIA Tesla P100 GPU with 16 GB memory for each experiment.

5.6.1 Empirical benchmarking of the tradeoff

We run Algorithm 5 for configurations I, III, IV and different powers of 2 as E. We use input strings sampled randomly from Σ^L. In order to characterize the time-memory tradeoff, we measure wall-clock time and peak GPU memory for a single gradient evaluation.

As discussed in Section 3.3, there are two methods to compute GA: the first (iterative) method doesn't compute and store the three-dimensional tensor $\mathbf{R}$ (3.7) explicitly resulting in a smaller memory consumption at the cost of less parallelization (onPS = False in Algorithm 3) while the second one computes tensor $\mathbf{R}$ using the parallel prefix sum algorithm, therefore operating faster but using more memory (onPS = True in Algorithm 3). The same methods can be applied for the memory-efficient algorithm when computing (5.4-5.6) updates. We implement and benchmark both methods as part of SLiM Performer. For the iterative algorithm, we implement its "block" version, when, instead of iterating i one-by-one, we iterate through blocks of a small size (see details in Appendix 5.B). This way the algorithm has a smaller constant in $O(L(M+d))$ time complexity and a bigger constant in the "small" $O(dM)$ term of the memory complexity (assuming that $d, M \ll L$).

For a fixed E, in addition to reporting memory of Algorithm 5, we also report memory of the naive gradient computation performed on a string of length E, sampled uniformly from Σ^E. This is to confirm that memory usage of Algorithm 5 is just slightly above the full computation on the input of length E.

Results are reported in Figure 5.3. We focus on configurations I, III and IV because they correspond to longer sequences. We observe significant improvements in memory compared to the full computation, as E decreases. As E converges to $2^0 = 1$, the remaining memory can be attributed to the storage of model's parameters $\boldsymbol{\theta}$. The time follows two regimes: fast decline as E grows (meaning that prefix sums are parallelized) and a slower decline for big values of E (meaning that the practical limit of parallelization is reached). We find that the iterative version of computing self-attention works only slightly slower than the prefix sum version while consuming much less memory. Finally, Algorithm 5 consumes only slightly more memory in practice than the full method run on the input of length $E < L$ ("L/B" plots in Figure 5.3, middle).

5.6.2 Comparison with checkpointing

In addition, we compare SLiM Performer with checkpointing (Griewank, 1992). By checkpointing in this context we understand storing $\{\mathbf{\Pi}^{(p)}\}_{1 \le p \le P}$ during the forward pass and reusing them during the backward pass instead of recomputing. This results in a small FLOPs decrease since $\mathbf{\Pi}^{(p)}$ are stored in memory, while the memory scales as $O(L/E)$. We use

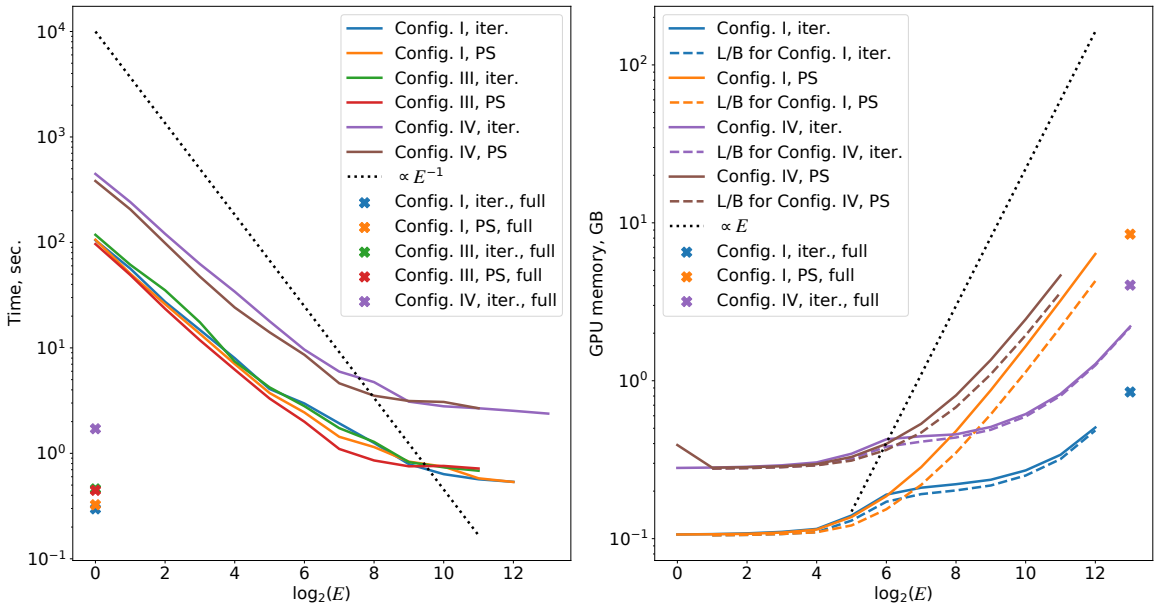


Fig. 5.3 Benchmarks of SLiM Performer. All plots are averaged over 10 seeds. "iter." stands for iterative computation of $\mathbf{R}$ while "PS" for explicit computation of $\mathbf{R}$. We omit time and memory for big values of E in "Config. IV, PS" and "Config. IV, full" setups, because these led to out-of-memory exceptions. **(Left)** Time dependence on E. Crosses indicate horizontal time levels for corresponding full memory-inefficient methods. The dotted line indicates $\propto E^{-1}$ tangent in logarithmic scale. **(Right)** Memory dependence (gigabytes) on E. Again, crosses are for horizontal levels of full sequence methods and the dotted line indicates $\propto E$ tangent. We do not report curves for the configuration III because they completely match curves for the configuration IV, which is natural since s, d_{hid} are the same for both configurations. "L/B" stands for a memory lower bound computed by processing input of length E.

configuration I for comparison (Figure 5.4). While not faster in practice, checkpointing consumes much more memory as E decreases for both iterative and prefix sum computations of $\mathbf{R}$.

5.6.3 Effects of finite-precision arithmetic

Since the iterative version results in a good balance between time and memory of Algorithm 5, we use it in our further experiments. To quantify finite-precision effects, we plot *relative discrepancy* $\|\nabla^{(E)}_{\boldsymbol{\theta}}\mathscr{L} - \nabla^{(\text{full})}_{\boldsymbol{\theta}}\mathscr{L}\| / \|\nabla^{(\text{full})}_{\boldsymbol{\theta}}\mathscr{L}\|$ between the gradient $\nabla^{(E)}_{\boldsymbol{\theta}}$ produced by Algorithm 5, and the gradient $\nabla^{(\text{full})}_{\boldsymbol{\theta}}\mathscr{L}$ produced by the full computation. Figure 5.5 shows results for randomly initialized models. We observe a small discrepancy of order 10^{-6}–10^{-4} confirming the correctness of Algorithm 5.

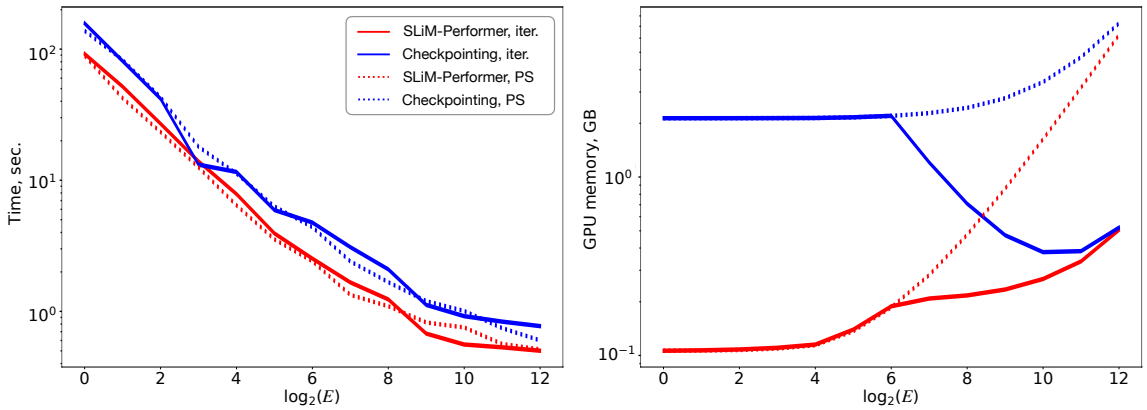


Fig. 5.4 SLiM Performer compared to checkpointing of $\{\mathbf{\Pi}^{(p)}\}_{1\leq p\leq P}$. Time and memory plots.

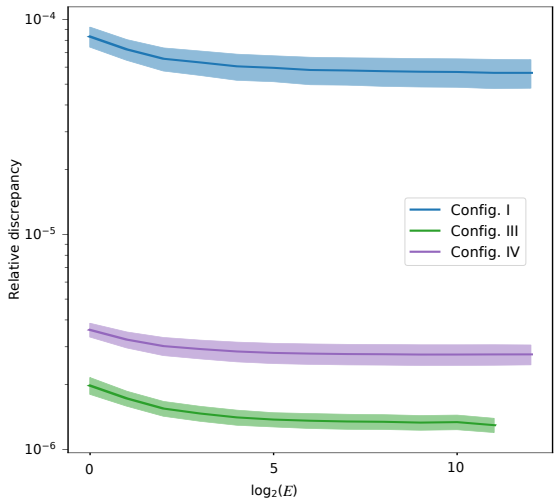


Fig. 5.5 Relative gradient discrepancy as a function of E and standard errors. Evaluated on random inputs over Σ^L.

5.6.4 Training from scratch and fine-tuning

To confirm backward compatibility of Algorithm 5 during training, we run three language modelling setups: copying task (CT), symbol-level Penn treebank (PTB) and Enwik8 (ENW).

For the CT, we follow the setup from (Katharopoulos et al., 2020; Kitaev et al., 2020), sampling inputs as $0\upsilon 0\upsilon$, where υ is drawn uniformly from $(\Sigma\setminus\{0\})^{L/2-1}$. In this setup, we only aggregate negative log-likelihood loss from the second half of the input, so the task is to reproduce the first half. We include the CT as a task where long range signal is crucial, and the heuristic of "chunking" the input into segments would fail to solve the task.

We use model configurations I, II, III for the CT, PTB and ENW resulting in sequence lengths $L = 8192, 1024, 4096$ respectively. For each setup we compare training with the full gradient computation and the "fine-tuning" regime when the first half of iterations is performed using the full algorithm, and the second half is run using Algorithm 5 with various values of E. In addition, we include training from scratch equipped with memory-efficient gradient computation via Algorithm 5. Figure 5.6 demonstrates results: all methods result in almost the same performance. Insignificant differences can be attributed to finite-precision arithmetic effects accumulating over many iterations. This confirms that memory-efficient gradient computation is backward-compatible during training. Table 5.2 quantifies the memory savings and time tradeoffs in all setups.

We use 200K, 100K, 200K SGD iterations in the copying task, Penn treebank and Enwik8 setups respectively. We use Adam optimizer (Kingma and Ba, 2015). For the copying task, we train with a learning rate 10^{-3} for 130K iterations and then decrease the learning rate to 10^{-4}. We use a fixed learning rate of 10^{-4} and 2×10^{-4} in Penn treebank and Enwik8 experiments respectively.

Table 5.2 Time per iteration (seconds, averaged over 1000 iterations) and peak GPU memory (gigabytes). When using small values of E, we relate the remaining memory to the storage of parameters $\boldsymbol{\theta}$ and the optimizer's state.

Setup, L, E	Time	Memory
CT, 8192, full	0.3008	0.938
CT, 8192, 4096	0.5372	0.595
CT, 8192, 2048	0.6002	**0.436**
PTB, 1024, full	0.1377	0.300
PTB, 1024, 512	0.2526	0.257
PTB, 1024, 256	0.3060	**0.231**
ENW, 4096, full	0.4598	1.513
ENW, 4096, 2048	0.7922	1.085
ENW, 4096, 1366	0.8654	**0.909**

5.6.5 One-shot fine-tuning under low memory

To analyze the scenario when model is pretrained on a server and then fine-tuned with the small E on a low-memory device, we add the following experiment. We take a pretrained model from either PTB or ENW setup from Section 5.6.4 and subsample randomly 5000

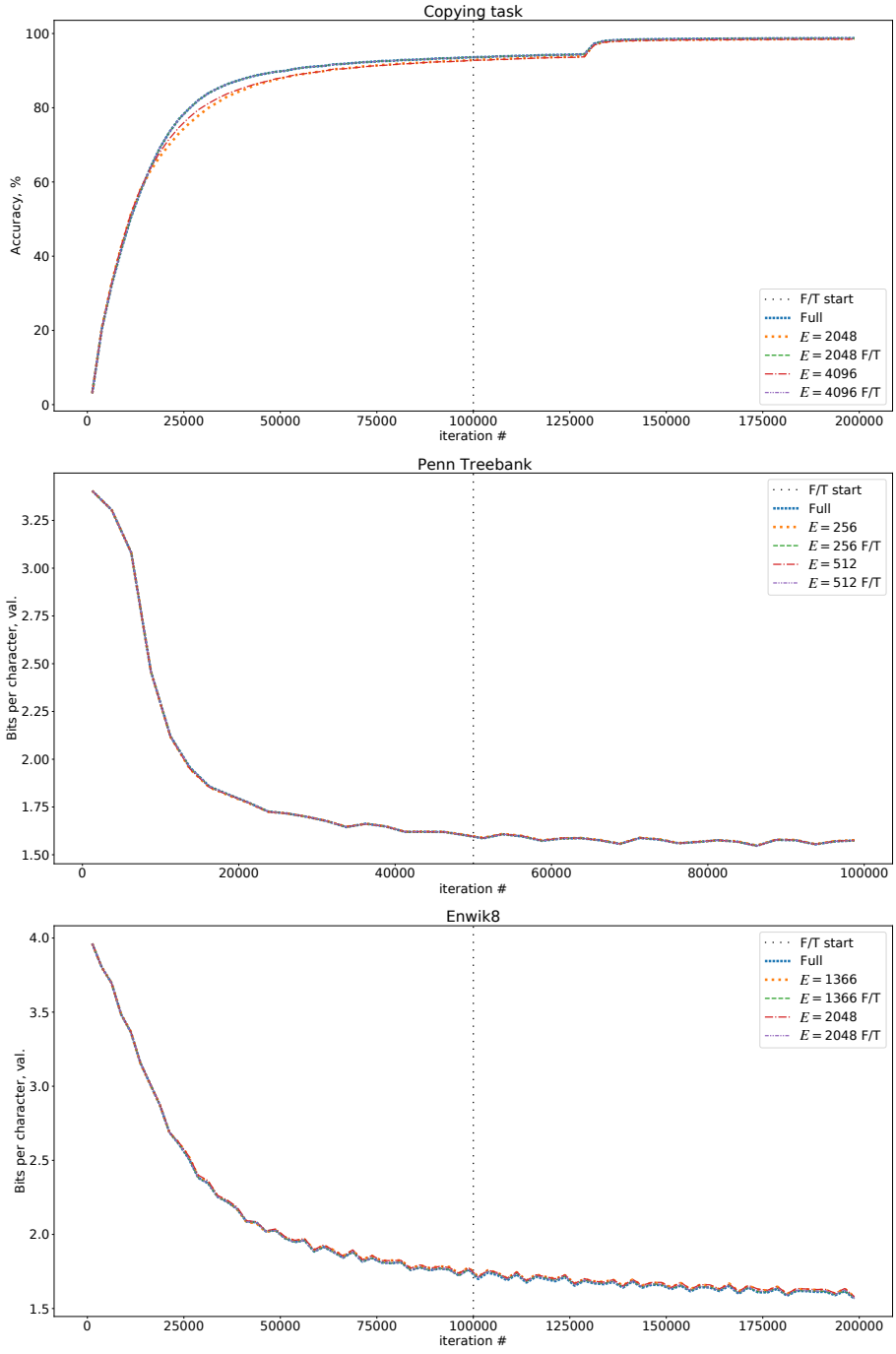


Fig. 5.6 Learning curves for three language modelling setups. We report accuracy on a newly generated data samples for the copying task, and the bits per character metric ($\mathscr{L}/\log 2$) on validation examples for Penn treebank and Enwik8. F/T stands for "fine-tuning".

examples from the corresponding validation set. We perform a one-step gradient descent with 0.01 learning rate (tuned on other random subset) to minimize the loss evaluated on

Table 5.3 Time per iteration (seconds, averaged over 1000 iterations) and peak GPU memory (gigabytes). When using small values of E, we relate the remaining memory to a storage of $\boldsymbol{\theta}$ and optimizer's state.

Setup, L, E	PTB, 1024, 16	ENW 4096, 64
Bits per character, fine-tuning	**1.4263**	**1.5642**
Bits per character, no fine-tuning	1.6544	1.6141
Fine-tuning memory, Algorithm 5	**0.1077**	**0.4276**
Fine-tuning memory, full computation	0.1834	0.8049
Fine-tuning time, Algorithm 5	0.8938	1.1142
Fine-tuning time, full computation	0.0426	0.2045

the first half of each sequence and evaluate bits per character on the second half. In this experiment the first half of the sequence represents the data generated by user on device, and the second half is a new data to be predicted. The fine-tuning procedure, therefore, represents a "personalization" of the model to a specific user (see Table 5.3). We observe the bits per character improvement without any server computations and memory improvement compared to the full computation while the time (≈ 1 seconds) is less crucial, since fine-tuning can run in the background.

5.7 Discussion

In this chapter we proposed SLiM Performer: an algorithm for computing forward and backward pass through the causal Performer with a linear $O(L)$ time and a sub-linear $O(E)$ memory complexity where E is a user defined integer smaller than L. Consequently, $E = 1$ results in the smallest $O(1)$ memory complexity. The algorithm is exact and doesn't involve any approximation, therefore it can be used with different values of E or even with the full Performer evaluation simultaneously. The algorithm can be used when even the batch size of 1 is expensive, i.e. when the gradient accumulation (Ott et al., 2018) technique cannot be used. This can be the case when one trains Transformers with many parameters and/or on long input sequences. Alternatively, the proposed algorithm can be used for fine-tuning on low-memory devices such as a smartphone. We presented an extensive empirical evaluation of SLiM Performer.

In the next Chapter, we will go deeper into random features and will find families of random features which extend TrigRFs and PosRFs. These families will be parameterized, and by optimizing the parameters we will be able to reduce variance of random features

while maintaining their positivity. This will result in a new generation of random-feature approximations of self-attention, FAVOR++, which has better theoretical concentration guarantees and which outperforms FAVOR+ in practice.

Appendix 5.A Proofs

Proof of Proposition 1

Proof. $\boldsymbol{\theta}^{(p)}$ doesn't affect terms $\mathscr{L}^{(1)}(\mathbf{X}^{(out,1)}),\ldots,\mathscr{L}^{(p-1)}(\mathbf{X}^{(out,n)})$, so the corresponding gradients are zero:

$$\nabla_{\boldsymbol{\theta}^{(p)}}\mathscr{L} = \nabla_{\boldsymbol{\theta}^{(p)}}\sum_{p'=p}^{P}\mathscr{L}^{(p')}(\mathbf{X}^{(out,p')}). \tag{5.13}$$

Similarly, $\boldsymbol{\Pi}^{(p)}$ does not affect $\mathscr{L}^{(1)},\ldots,\mathscr{L}^{(p)}$, so

$$\boldsymbol{g}\boldsymbol{\Pi}^{(p)} = \nabla_{\boldsymbol{\Pi}^{(p)}}\mathscr{L} = \nabla_{\boldsymbol{\Pi}^{(p)}}\sum_{p'=p+1}^{P}\mathscr{L}^{(p)}(\mathbf{X}^{(out,p')}). \tag{5.14}$$

In particular,

$$\boldsymbol{g}\boldsymbol{\Pi}^{(P)} = \nabla_{\boldsymbol{\Pi}^{(P)}}\mathscr{L} = \mathbf{0}_{r\times D_1}. \tag{5.15}$$

For all $1 \le p < p' \le P$, $\boldsymbol{\theta}^{(p)}$ and $\boldsymbol{\Pi}^{(p-1)}$ affect $\mathscr{L}^{(p')}$ only through $\boldsymbol{\Pi}^{(p)}$, so according to the chain rule,

$$\begin{aligned}\nabla_{\boldsymbol{\theta}^{(p)}}\sum_{p'=p+1}^{P}\mathscr{L}^{(p')}(\mathbf{X}^{(out,n')}) &= \sum_{r=1}^{s}\left(\frac{\partial\boldsymbol{\Pi}_r^{(p)}}{\partial\boldsymbol{\theta}^{(p)}}\right)^{\top}\times\nabla_{\boldsymbol{\Pi}_r^{(p)}}\sum_{p'=p+1}^{P}\mathscr{L}^{(p')}(\mathbf{X}^{(out,p')})\\ &= \sum_{r=1}^{s}\frac{\partial\boldsymbol{\Pi}_r^{(p)}}{\partial\boldsymbol{\theta}^{(p)}}^{\top}\times\nabla_{\boldsymbol{\Pi}_r^{(p)}}\mathscr{L},\end{aligned} \tag{5.16}$$

where $\frac{\partial\square}{\partial\square}$ denotes a Jacobian matrix between flattened $\square$'s and we use (5.14) in the second transition. Similarly, for all $1 \le r' \le s$

$$\begin{aligned}\nabla_{\boldsymbol{\Pi}_{r'}^{(p-1)}}\sum_{p'=p+1}^{P}\mathscr{L}^{(p')}(\mathbf{X}^{(out,p')}) &= \sum_{r=1}^{s}\left(\frac{\partial\boldsymbol{\Pi}_r^{(p)}}{\partial\boldsymbol{\Pi}_{r'}^{(p-1)}}\right)^{\top}\times\nabla_{\boldsymbol{\Pi}_r^{(p)}}\sum_{p'=p+1}^{P}\mathscr{L}^{(p')}(\mathbf{X}^{(out,p')})\\ &= \sum_{r=1}^{s}\frac{\partial\boldsymbol{\Pi}_r^{(p)}}{\partial\boldsymbol{\Pi}_{r'}^{(p-1)}}^{\top}\times\nabla_{\boldsymbol{\Pi}_r^{(p)}}\mathscr{L}.\end{aligned}$$

Further, it's easy to see that for all $1 \leq r \leq s$:

$$\left(\frac{\partial \mathbf{\Pi}_r^{(p)}}{\partial \square}\right)^\top \times \nabla_{\mathbf{\Pi}_r^{(p)}} \mathscr{L} = \nabla_\square \left([\mathbf{\Pi}_r^{(p)}]^\top \langle\langle \nabla_{\mathbf{\Pi}_r^{(p)}} \mathscr{L} \rangle\rangle \right), \tag{5.17}$$

where $\square \in \{\boldsymbol{\theta}^{(p)}\} \cup \{\mathbf{\Pi}_{r'}^{(p-1)}\}_{1 \leq r' \leq s}$. $\langle\langle \cdot \rangle\rangle$ denotes a *stop-gradient* operation, i.e. gradients are not propagated inside brackets and the argument is considered as a constant.

We conclude that

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}^{(p)}} \mathscr{L} = \nabla_{\boldsymbol{\theta}^{(p)}} \mathscr{L}^{(p)}(\mathbf{X}^{(out,p)}) + \nabla_{\boldsymbol{\theta}^{(p)}} \sum_{p'=p+1}^{P} \mathscr{L}^{(p')}(\mathbf{X}^{(out,p')}) = \nabla_{\boldsymbol{\theta}^{(p)}} \mathscr{L}^{(p)}(\mathbf{X}^{(out,p)}) \\
+ \sum_{r=1}^{s} \left(\frac{\partial \mathbf{\Pi}_r^{(p)}}{\partial \boldsymbol{\theta}^{(p)}}\right)^\top \times \nabla_{\mathbf{\Pi}_r^{(p)}} \mathscr{L} \\
= \nabla_{\boldsymbol{\theta}^{(p)}} \left(\mathscr{L}^{(p)}(\mathbf{X}^{(out,p)}) + \sum_{r=1}^{s} [\mathbf{\Pi}_r^{(p)}]^\top \langle\langle \nabla_{\mathbf{\Pi}_r^{(p)}} \mathscr{L} \rangle\rangle \right) = \nabla_{\boldsymbol{\theta}^{(p)}} \Phi^{(p)}(\boldsymbol{\theta}^{(p)}, \mathbf{\Pi}^{(p-1)}, \nabla_{\mathbf{\Pi}^{(p)}} \mathscr{L}) \\
= \nabla_{\boldsymbol{\theta}^{(p)}} \Phi^{(p)}(\boldsymbol{\theta}^{(p)}, \mathbf{\Pi}^{(p-1)}, \boldsymbol{g}\mathbf{\Pi}^{(p)})
\end{aligned} \tag{5.18}$$

where the first transition is due to (5.13), the second is due to (5.16), the third is due to (5.17), the fourth and the fifth are by the definition of $\Phi^{(p)}$ and $\boldsymbol{g}\mathbf{\Pi}^{(p)}$ respectively. Similarly, for all $1 \leq r' \leq s$:

$$\begin{aligned}
\boldsymbol{g}\mathbf{\Pi}_{r'}^{(p-1)} = \nabla_{\mathbf{\Pi}_{r'}^{(p-1)}} \mathscr{L} = \nabla_{\mathbf{\Pi}_{r'}^{(p-1)}} \mathscr{L}^{(p)}(\mathbf{X}^{(out,p)}) + \nabla_{\mathbf{\Pi}_{r'}^{(p-1)}} \sum_{p'=p+1}^{P} \mathscr{L}^{(p')}(\mathbf{X}^{(out,p')}) \\
= \nabla_{\mathbf{\Pi}_{r'}^{(p-1)}} \mathscr{L}^{(p)}(\mathbf{X}^{(out,p)}) + \sum_{r=1}^{s} \frac{\partial \mathbf{\Pi}_r^{(p)}}{\partial \mathbf{\Pi}_{r'}^{(p-1)}}^\top \times \nabla_{\mathbf{\Pi}_r^{(p)}} \mathscr{L} \\
= \nabla_{\mathbf{\Pi}_{r'}^{(p-1)}} \left(\mathscr{L}^{(p)}(\mathbf{X}^{(out,p)}) + \sum_{r=1}^{s} \nabla_\square [\mathbf{\Pi}_r^{(p)}]^\top \langle\langle \nabla_{\mathbf{\Pi}_r^{(p)}} \mathscr{L} \rangle\rangle \right) \\
= \nabla_{\mathbf{\Pi}_{r'}^{(p-1)}} \Phi^{(p)}(\boldsymbol{\theta}^{(p)}, \mathbf{\Pi}^{(p-1)}, \nabla_{\mathbf{\Pi}^{(p)}} \mathscr{L}) = \nabla_{\mathbf{\Pi}_{r'}^{(p-1)}} \Phi^{(p)}(\boldsymbol{\theta}^{(p)}, \mathbf{\Pi}^{(p-1)}, \boldsymbol{g}\mathbf{\Pi}^{(p)}),
\end{aligned} \tag{5.19}$$

where the first transition is by the definition of $\boldsymbol{g}\mathbf{\Pi}^{(p-1)}$, the second is due to (5.14), the third is due to (5.16), the fourth is due to (5.17), the fifth and the sixth are by the definition of $\Phi^{(p)}$ and $\boldsymbol{g}\mathbf{\Pi}^{(p)}$ respectively. Since (5.19) holds for all rows of $\boldsymbol{g}\mathbf{\Pi}^{(p-1)}$, we conclude that

$$\boldsymbol{g}\mathbf{\Pi}^{(p-1)} = \nabla_{\mathbf{\Pi}^{(p-1)}} \Phi^{(p)}(\boldsymbol{\theta}^{(p)}, \mathbf{\Pi}^{(p-1)}, \boldsymbol{g}\mathbf{\Pi}^{(p)}). \tag{5.20}$$

(5.15,5.18,5.20) conclude the proof of the proposition. □

Appendix 5.B Efficient "block" computation of (3.8)

According to Algorithm 3, (3.8) is computed as follows by setting onGA = onCausal = True, onFullPS = False. Compute $\mathbf{P}^{(1,\mathrm{GA})}, \mathbf{P}^{(2,\mathrm{GA})}$ as defined in (3.9) and set $\mathbf{C} = \begin{bmatrix}\mathbf{V} & \mathbf{1}_L\end{bmatrix}$. Initialize $\mathbf{R}^{\mathrm{cur}} = \mathbf{0}_{(d+1)\times M}$, iterate over $i := 1, \ldots, L$ and compute

$$\begin{aligned}
&\mathbf{R}^{\mathrm{cur}} := \mathbf{R}^{\mathrm{cur}} + \mathbf{C}_i \times \left(\mathbf{P}_i^{(2,GA)}\right)^\top;\\
&\mathbf{Buf}_i^{(2)} := \mathbf{R}^{\mathrm{cur}} \times \mathbf{P}_i^{(1,GA)};\\
&\mathbf{Y}_l := \left(\mathbf{Buf}_{i,d+1}^{(2)}\right)^{-1} \mathbf{Buf}_{i,:d}^{(2)}.
\end{aligned}$$

This way, the 3D tensor $\mathbf{R} \in \mathbb{R}^{L\times(d+1)\times M}$ is not stored in memory explicitly, resulting in $O(L)$ time and $O(L(d+M)+dM)$ memory complexity. In order to have the same memory consumption during back-propagation, (Katharopoulos et al., 2020) propose the following routine. Keep the buffer $\mathbf{R}^{\mathrm{cur}}$ as the result of the forward pass and initialize a gradient buffer $\nabla_{\mathbf{R}^{\mathrm{cur}}}\mathscr{L} = \mathbf{0}_{(d+1)\times M}$. Assuming that $\nabla_{\mathbf{Buf}^{(2)}}\mathscr{L} \in \mathbb{R}^{L\times(d+1)}$ is computed using automatic differentiation (Griewank and Walther, 2008) from $\nabla_{\mathbf{Y}}\mathscr{L}$, iterate in the backward direction $i := L, \ldots, 1$ and compute

$$\begin{aligned}
&\left(\nabla_{\mathbf{P}^{(1,GA)}}\mathscr{L}\right)_i = \left(\mathbf{R}^{\mathrm{cur}}\right)^\top \times \left(\nabla_{\mathbf{Buf}^{(2)}}\mathscr{L}\right)_i;\\
&\mathbf{R}^{\mathrm{cur}} := \mathbf{R}^{\mathrm{cur}} - \mathbf{C}_i \times \left(\mathbf{P}_i^{(2,GA)}\right)^\top;\\
&\nabla_{\mathbf{R}^{\mathrm{cur}}}\mathscr{L} := \nabla_{\mathbf{R}^{\mathrm{cur}}}\mathscr{L} + \left(\nabla_{\mathbf{Buf}^{(2)}}\mathscr{L}\right)_i \times \left(\mathbf{P}_i^{(1,GA)}\right)^\top;\\
&\left(\nabla_{\mathbf{V}}\mathscr{L}\right)_i := \left(\nabla_{\mathbf{R}^{\mathrm{cur}}}\mathscr{L}\right)_{:d} \times \mathbf{P}_i^{(2,GA)};\\
&\left(\nabla_{\mathbf{P}^{(2,GA)}}\mathscr{L}\right)_i = \left(\nabla_{\mathbf{R}^{\mathrm{cur}}}\mathscr{L}\right)^\top \times \mathbf{C}_i.
\end{aligned}$$

This way, we can compute $\nabla_{\mathbf{V}}\mathscr{L}$, $\nabla_{\mathbf{P}^{(1,GA)}}\mathscr{L}$, $\nabla_{\mathbf{P}^{(2,GA)}}\mathscr{L}$ without storing the whole tensor $\mathbf{R}$ and its gradient in memory.

In practice, the described forward-backward algorithm works slowly when implemented in automatic differentiation libraries for GPUs such as Tensorflow (Abadi et al., 2015) or PyTorch (Paszke et al., 2017), because i is iterated one-by-one: (Katharopoulos et al., 2020) use low-level CUDA extensions to make the algorithm practical. Instead, we propose a "block" version, when we iterate through blocks of i of a small size $\mathscr{B}$ (we use $\mathscr{B} = 64$). In each block, we use built-in hardware-optimized prefix sum functions implemented in the library. We use them on the inputs of length $\mathscr{B}$ to find $\mathbf{Y}_{i:i+\mathscr{B}-1}$ using the maintained front $\mathbf{R}^{\mathrm{cur}}$. The formal algorithm is as follows. Initialize a buffer $\mathbf{R}^{\mathrm{cur}} = \mathbf{0}_{(d+1)\times M}$. For simplicity

assuming that $\mathscr{B}$ divides L (extension for the opposite case is straightforward), iterate over $i = 1, \mathscr{B}+1, \ldots, L-\mathscr{B}+1$ and compute

$$\begin{aligned}
&\mathbf{R}^{\text{block}} := \text{PS}\left(\left(\mathbf{C}_{i+i'-1} \times \left(\mathbf{P}_{i+i'-1}^{(2,GA)}\right)^\top\right)_{i'=1}^{\mathscr{B}}\right) \in \mathbb{R}^{\mathscr{B}\times(d+1)\times M}; \qquad (5.21)\\
&\mathbf{R}^{\text{block}} := \left(\mathbf{R}^{\text{cur}} + \mathbf{R}_{i'}^{\text{block}}\right)_{i'=1}^{\mathscr{B}};\\
&\mathbf{R}^{\text{cur}} := \mathbf{R}_{\mathscr{B}}^{\text{block}};\\
&\mathbf{Buf}_{i:i+\mathscr{B}-1}^{(2)} := \left(\mathbf{R}_{i'}^{\text{block}} \times \mathbf{P}_{i'}^{(1,GA)}\right)_{i'=1}^{\mathscr{B}};\\
&\mathbf{Y}_{i:i+\mathscr{B}-1} := \left(\left(\mathbf{Buf}_{i+i'-1,d+1}^{(2)}\right)^{-1} \mathbf{Buf}_{i+i'-1,:d}^{(2)}\right)_{i'=1}^{\mathscr{B}}.
\end{aligned}$$

In the "block" version, the number of outer sequential iterations is reduced to $L/\mathscr{B}$, resulting in $O((L/\mathscr{B})\log\mathscr{B})$ parallel time complexity when the logarithmic parallel algorithm is used to compute the prefix sum (5.21). The memory complexity of the algorithm is $O(L(d+M)+\mathscr{B}dM)$, where the second term is for storing $\mathbf{R}^{\text{block}}$. Assuming that $\mathscr{B}$ is a small constant ($\mathscr{B} = O(1)$), we conclude that the "block" version has $O(L(d+M)+dM)$ memory and $O(L)$ time complexity – same as the algorithm of (Katharopoulos et al., 2020). As for hidden constants in complexity estimates, the constant inside $O(L)$ time complexity is reduced at the cost of increasing the constant of the "small" dM term in the memory complexity (when $d, M \ll L$), making the "block" iterative algorithm a practical choice for computing causal GA (3.8).

We further show how to back-propagate through the causal GA (3.8) in $O((L/\mathscr{B})\log\mathscr{B})$ time and $O(L(d+M)+\mathscr{B}dM)$ memory. Again, keep the buffer $\mathbf{R}^{\text{cur}}$ as the result of forward pass, and initialize the gradient buffer $\nabla_{\mathbf{R}^{\text{cur}}}\mathscr{L} = \mathbf{0}_{(d+1)\times M}$. Assuming that $\nabla_{\mathbf{Buf}^{(2)}}\mathscr{L} \in \mathbb{R}^{L\times(d+1)}$ is computed using automatic differentiation (Griewank and Walther, 2008), iterate in a backward direction $i = L-\mathscr{B}+1, L-2\mathscr{B}+1, \ldots, 1$ and compute

$$\begin{aligned}
&\mathbf{R}^{\text{cur}} := \mathbf{R}^{\text{cur}} - \sum_{i'=i}^{i+\mathscr{B}-1} \mathbf{C}_i \times \left(\mathbf{P}_i^{(2,GA)}\right)^\top;\\
&\mathbf{R}^{\text{block}} := \text{PS}\left(\left(\mathbf{C}_{i+i'-1} \times \left(\mathbf{P}_{i+i'-1}^{(2,GA)}\right)^\top\right)_{i'=1}^{\mathscr{B}}\right);\\
&\mathbf{R}^{\text{block}} := \left(\mathbf{R}^{\text{cur}} + \mathbf{R}_{i'}^{\text{block}}\right)_{i'=1}^{\mathscr{B}};\\
&\left(\nabla_{\mathbf{P}^{(1,GA)}}\mathscr{L}\right)_{i:i+\mathscr{B}-1} = \left(\left(\mathbf{R}_{i'}^{\text{block}}\right)^\top \times \left(\nabla_{\mathbf{Buf}^{(2)}}\mathscr{L}\right)_{i+i'-1}\right)_{i'=1}^{\mathscr{B}};
\end{aligned}$$

$$\nabla_{\mathbf{R}^{\text{cur}}}\mathscr{L} := \nabla_{\mathbf{R}^{\text{cur}}}\mathscr{L} + \sum_{i'=i}^{i+\mathscr{B}-1} \left(\nabla_{\mathbf{Buf}^{(2)}}\mathscr{L}\right)_i \times \left(\mathbf{P}_i^{(1,GA)}\right)^\top;$$

$$\nabla_{\mathbf{R}^{\text{block}}}\mathscr{L} := \text{PS}\left(\left(\left(\nabla_{\mathbf{Buf}^{(2)}}\mathscr{L}\right)_{i+i'-1} \times \left(\mathbf{P}_{i+i'-1}^{(1,GA)}\right)^\top\right)_{i'=1}^{\mathscr{B}}\right);$$

$$\nabla_{\mathbf{R}^{\text{block}}}\mathscr{L} := \left(\nabla_{\mathbf{R}^{\text{cur}}}\mathscr{L} - \left(\nabla_{\mathbf{R}^{\text{block}}}\mathscr{L}\right)_{i'}\right)_{i'=1}^{\mathscr{B}};$$

$$\left(\nabla_{\mathbf{V}}\mathscr{L}\right)_{i:i+\mathscr{B}-1} := \left(\left(\nabla_{\mathbf{R}^{\text{block}}}\mathscr{L}\right)_{i',:d} \times \mathbf{P}_{i+i'-1}^{(2,GA)}\right)_{i'=1}^{\mathscr{B}};$$

$$\left(\nabla_{\mathbf{P}^{(2,GA)}}\mathscr{L}\right)_{i:i+\mathscr{B}-1} = \left(\left(\nabla_{\mathbf{R}^{\text{block}}}\mathscr{L}\right)_{i',:d}^\top \times \mathbf{C}_{i+i'-1}\right)_{i'=1}^{\mathscr{B}}.$$

Finally, it's easy to see how to use both one-to-one and "block" iterative computation as part of Algorithm 5 to compute the update (5.4-5.6). For that, when doing a forward computation for some p, r, initialize $\mathbf{R}^{\text{cur}}$ from the corresponding sub-vector of $\boldsymbol{g}\boldsymbol{\Pi}_r^{(p)}$, with the rest of the algorithm unchanged. Similarly, during a backward pass for some p, r, initialize $\nabla_{\mathbf{R}^{\text{cur}}}\mathscr{L}$ from the corresponding sub-vector of $\boldsymbol{g}\boldsymbol{\Pi}_r^{(p)}$ and leave the rest of the iterative back-propagation algorithm unchanged.

Chapter 6

Chef's random tables: going deeper into random features

6.1 Motivation

By this point, we have discussed two types of random features for the Gaussian kernel (Definition 1): trigonometric random features (TrigRFs) defined in Section 2.1.3 which have a form

$$f^{(1)}_{\text{trig}}(\boldsymbol{\omega}, \mathbf{x}) = \exp(\mathrm{i}\boldsymbol{\omega}^\top \mathbf{x}), \quad f^{(2)}_{\text{trig}}(\boldsymbol{\omega}, \mathbf{y}) = \exp(-\mathrm{i}\boldsymbol{\omega}^\top \mathbf{y}) \tag{6.1}$$

and positive random features (PosRFs) defined in Section 4.2 which have a form

$$f^{(1)}_{\text{trig}}(\boldsymbol{\omega}, \mathbf{x}) = \exp(\boldsymbol{\omega}^\top \mathbf{x} - \|\mathbf{x}\|^2), \quad f^{(2)}_{\text{trig}}(\boldsymbol{\omega}, \mathbf{y}) = \exp(\boldsymbol{\omega}^\top \mathbf{y} - \|\mathbf{y}\|^2). \tag{6.2}$$

In both variants, the $\boldsymbol{\omega}$'s distribution is the same: standard multivariate Gaussian $p_{\text{SG}}(\boldsymbol{\omega})$. Both types of features have disadvantages. TrigRFs can take negative values which results in unstable training and nonpositive self-attention weights, as discussed in Section 4.1. Positive random features don't have this problem but they are unbounded, meaning that they can be arbitrarily big and, hence, it's unclear how to obtain concentration results similar to Theorem 3. For this reason we proposed and analysed its regularized modification in Section 4.3.

TrigRFs (6.1) and PosRFs (6.2) have surprisingly similar expressions, and the natural question is whether we can propose a generalization of both mechanisms which would interpolate or extend both of them and would be free of the aforementioned drawbacks. This chapter is dedicated to the search of such an ultimate random feature mechanism. First, we consider an interpolation or, as we call it, a *hybrid* of both random feature types which has

a form of linear combination of both where the weights are defined via the angle between vectors $\mathbf{x}, \mathbf{y}$.

After that we discover a whole new family of *generalized exponential* random features which extend both TrigRFs (6.1) and PosRFs (6.2). This family is parametrized and its variance can be reduced by optimizing its parameters. Further, the parameters can be restricted in such a way that these random features stay positive-valued. It appears that a close form solution for optimal parameters exists in this case, and we refer to the resulting mechanism as *optimal positive random features*. After that, we discover a completely different family of random features for the Gaussian kernel which is based on the unbiased approximation of Taylor series and is induced by discrete random vectors $\boldsymbol{\omega}$ (*discretely-induced random features*). We refer to all these different non-hybrid variants as *chef's random tables (CRTs)* in analogy to *random kitchen sinks* from the original paper trilogy on random features for the Gaussian kernel (Rahimi and Recht, 2007, 2008a,b).

We observe that the optimal positive mechanism resolves disadvantages of earlier variants: it is both positive-valued and bounded. Because of that, we are able to derive exponentially fast concentration bounds similar to Theorem 3 for this method. Further, we observe that this method performs better empirically compared to hybrid features and other methods from CRTs. This leads to emergence of *FAVOR++*: a new iteration of random feature mechanisms for self-attention approximation. We evaluate this mechanism in large scale Transformer training setups and conclude that it consistently outperforms FAVOR+.

The chapter is structured as follows:

- Section 6.2 introduces hybrid random features: a linear combination of TrigRFs and PosRFs which is aimed to reduce variance of random features when the angle between $\mathbf{x}$ and $\mathbf{y}$ is close to 0 and π. The multipliers are chosen using the random feature angle estimate (Choromanski et al., 2017b).

- Section 6.3 is about generalized exponential random features: a unified form of both TrigRFs and PosRFs. The variance of these random features can be minimized constrained to the features staying positive-valued which gives a birth to optimal positive random features.

- Section 6.4 proposes discretely induced random features based on the stochastic approximation of exponent's Taylor expansion. Depending on the distribution used for the approximation, we obtain special instantiations: *Poisson random features* and *geometric random features*.

- Section 6.5 is dedicated to a theoretical analysis of generalized exponential and optimal positive random features. Since, similarly to TrigRFs and PosRFs, these types are

induced by the standard multivariate Gaussian distribution $p_{\mathrm{SG}}(\boldsymbol{\omega})$, a natural variance reduction technique is to use orthogonal Gaussian $\boldsymbol{\omega}$'s as in Sections 2.1.4 and 4.2. We obtain variance reduction results similar to Theorems 3 and 6.

- Section 6.6 is dedicated to a thorough empirical evaluation of all new random feature variants. We compare all new variants in the task of nonparametric classification and deduce that the optimal positive variant performs best. Since it's also positive-valued, we take it as a base of a new FAVOR++ mechanism for self-attention approximation using orthogonal random features. We evaluate this mechanism in a number of real life Transformer setups and conclude that this method outperforms FAVOR+ in all tasks.
- Section 6.7 is reserved for concluding discussions.

6.2 Hybrid random features

As we observed in Figure 4.1, variance of TrigRFs and PosRFs becomes zero when the angle θ between $\mathbf{x}$ and $\mathbf{y}$ is 0 and π respectively. It would be good to come up with a combination of both TrigRFs and PosRFs which would resemble TrigRFs when θ is close to 0 and PosRFs when it's close to π.

More technically, fix nonzero $\mathbf{x},\mathbf{y}\in\mathbb{R}^d$ and let $\theta=\arccos\left(\frac{\mathbf{x}^\top\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}\right)$ be the angle between $\mathbf{x}$ and $\mathbf{y}$. Let $\widehat{\mathbf{x}}_{\mathrm{trig}},\widehat{\mathbf{y}}_{\mathrm{trig}}$ be defined as

$$\widehat{\mathbf{x}}_{\mathrm{trig}}=M_1^{-1/2}(f^{(1)}_{\mathrm{trig}}(\boldsymbol{\omega}^{(1,m)},\mathbf{x}))_{m=1}^{M_1},\quad \widehat{\mathbf{y}}_{\mathrm{trig}}=M_1^{-1/2}(f^{(2)}_{\mathrm{trig}}(\boldsymbol{\omega}^{(1,m)},\mathbf{y}))_{m=1}^{M_1}$$

where $M_1\in\mathbb{N}$, $\boldsymbol{\omega}^{(1,1)},\ldots,\boldsymbol{\omega}^{(1,M_1)}\sim\mathcal{N}(0,1)^d$ are i.i.d. Further, let $\widehat{\mathbf{x}}_{\mathrm{pos}},\widehat{\mathbf{y}}_{\mathrm{pos}}$ be defined as

$$\widehat{\mathbf{x}}_{\mathrm{pos}}=M_2^{-1/2}(f^{(1)}_{\mathrm{pos}}(\boldsymbol{\omega}^{(2,m)},\mathbf{x}))_{m=1}^{M_2},\quad \widehat{\mathbf{y}}_{\mathrm{pos}}=M_2^{-1/2}(f^{(2)}_{\mathrm{pos}}(\boldsymbol{\omega}^{(2,m)},\mathbf{y}))_{m=1}^{M_2}$$

where $M_2\in\mathbb{N}$, $\boldsymbol{\omega}^{(2,1)},\ldots,\boldsymbol{\omega}^{(2,M_2)}\sim\mathcal{N}(0,1)^d$ are i.i.d. Then, as discussed in Sections 2.1.3 and 4.2, $\mathrm{Re}\left(\widehat{\mathbf{x}}^\top_{\mathrm{trig}}\widehat{\mathbf{y}}_{\mathrm{trig}}\right)$ and $\widehat{\mathbf{x}}^\top_{\mathrm{pos}}\widehat{\mathbf{y}}_{\mathrm{pos}}$ are unbiased estimators of $K(\mathbf{x},\mathbf{y})$. Clearly, the following estimator of $K(\mathbf{x},\mathbf{y})$ is also unbiased:

$$\left(1-\frac{\theta}{\pi}\right)\mathrm{Re}\left(\widehat{\mathbf{x}}^\top_{\mathrm{trig}}\widehat{\mathbf{y}}_{\mathrm{trig}}\right)+\frac{\theta}{\pi}\widehat{\mathbf{x}}^\top_{\mathrm{pos}}\widehat{\mathbf{y}}_{\mathrm{pos}}. \tag{6.3}$$

This estimator approaches TrigRF estimator $\mathrm{Re}\left(\widehat{\mathbf{x}}^\top_{\mathrm{trig}}\widehat{\mathbf{y}}_{\mathrm{trig}}\right)$ when $\theta\to 0$ and PosRF estimator $\widehat{\mathbf{x}}^\top_{\mathrm{pos}}\widehat{\mathbf{y}}_{\mathrm{pos}}$ when $\theta\to\pi$, hence it will have low variance in both cases. However, it's unclear

how to represent this estimator in the random feature form and use it for the Gaussian kernel matrix approximation as in Section 2.1.2.

It appears that there exists a random feature decomposition of the angle θ. Namely, define

$$\widehat{\mathbf{x}}_{\text{ang}} = M_3^{-1/2}(\text{sign}(\mathbf{x}^\top \boldsymbol{\omega}^{(3,m)}))_{m=1}^{M_3}, \quad \widehat{\mathbf{y}}_{\text{ang}} = M_3^{-1/2}(\text{sign}(\mathbf{y}^\top \boldsymbol{\omega}^{(3,m)}))_{m=1}^{M_3}$$

where $M_3 \in \mathbb{N}$, $\boldsymbol{\omega}^{(3,1)}, \ldots, \boldsymbol{\omega}^{(3,M_3)} \sim \mathcal{N}(0,1)^d$ are i.i.d. Then, $\widehat{\mathbf{x}}_{\text{ang}}^\top \widehat{\mathbf{y}}_{\text{ang}}$ is an unbiased estimator of $1 - 2\theta/\pi$: $\mathbb{E}\widehat{\mathbf{x}}_{\text{ang}}^\top \widehat{\mathbf{y}}_{\text{ang}} = 1 - 2\theta/\pi$ (Choromanski et al., 2017b). Assume that $\{\boldsymbol{\omega}^{(3,m)}\}$ are sampled independently from $\{\boldsymbol{\omega}^{(1,m)}\}$ and $\{\boldsymbol{\omega}^{(2,m)}\}$. Then we substitute the estimator (6.3) by the following new estimator:

$$\frac{1}{2}\left(1 + \widehat{\mathbf{x}}_{\text{ang}}^\top \widehat{\mathbf{y}}_{\text{ang}}\right) \text{Re}\left(\widehat{\mathbf{x}}_{\text{trig}}^\top \widehat{\mathbf{y}}_{\text{trig}}\right) + \frac{1}{2}\left(1 - \widehat{\mathbf{x}}_{\text{ang}}^\top \widehat{\mathbf{y}}_{\text{ang}}\right) \widehat{\mathbf{x}}_{\text{pos}}^\top \widehat{\mathbf{y}}_{\text{pos}}. \tag{6.4}$$

This is an unbiased estimator of $K(\mathbf{x}, \mathbf{y})$ since

$$\begin{aligned}
&\mathbb{E}\left(\frac{1}{2}\left(1 + \widehat{\mathbf{x}}_{\text{ang}}^\top \widehat{\mathbf{y}}_{\text{ang}}\right) \text{Re}\left(\widehat{\mathbf{x}}_{\text{trig}}^\top \widehat{\mathbf{y}}_{\text{trig}}\right) + \frac{1}{2}\left(1 - \widehat{\mathbf{x}}_{\text{ang}}^\top \widehat{\mathbf{y}}_{\text{ang}}\right) \widehat{\mathbf{x}}_{\text{pos}}^\top \widehat{\mathbf{y}}_{\text{pos}}\right) \\
&= \mathbb{E}\left(\frac{1}{2}\left(1 + \widehat{\mathbf{x}}_{\text{ang}}^\top \widehat{\mathbf{y}}_{\text{ang}}\right)\right) \mathbb{E}\left(\text{Re}\left(\widehat{\mathbf{x}}_{\text{trig}}^\top \widehat{\mathbf{y}}_{\text{trig}}\right)\right) + \mathbb{E}\left(\frac{1}{2}\left(1 - \widehat{\mathbf{x}}_{\text{ang}}^\top \widehat{\mathbf{y}}_{\text{ang}}\right)\right) \mathbb{E}\left(\widehat{\mathbf{x}}_{\text{pos}}^\top \widehat{\mathbf{y}}_{\text{pos}}\right) \\
&= \left(1 - \frac{\theta}{\pi}\right) K(\mathbf{x}, \mathbf{y}) + \frac{\theta}{\pi} K(\mathbf{x}, \mathbf{y}) = K(\mathbf{x}, \mathbf{y}).
\end{aligned}$$

Interestingly, the variance of the angle estimator (Choromanski et al., 2017b)

$$\text{Var}\, \widehat{\mathbf{x}}_{\text{ang}}^\top \widehat{\mathbf{y}}_{\text{ang}} = \frac{4\theta(\pi - \theta)}{M_3 \pi^2} \tag{6.5}$$

is zero when $\theta \in \{0, \pi\}$. That is, the angle estimator becomes exact when θ approaches 0 or π which means that (6.4) has the same properties as (6.3): it also becomes exact when θ approaches 0 or π. We refer to the Gaussian kernel estimator of type (6.4) as a *hybrid random feature (HybRF) estimator*. An important property of (6.4) is that it can be expressed in the random feature form which is compatible with the efficient approximation of the Gaussian kernel matrix. Consider the following complex-valued random vectors $\widehat{\mathbf{x}}_{\text{hyb}}, \widehat{\mathbf{y}}_{\text{hyb}}$:

$$\widehat{\mathbf{x}}_{\text{hyb}} = \frac{1}{\sqrt{2}}\left[\widehat{\mathbf{x}}_{\text{trig}}^\top \quad \text{fl}(\widehat{\mathbf{x}}_{\text{ang}} \widehat{\mathbf{x}}_{\text{trig}}^\top)^\top \quad \widehat{\mathbf{x}}_{\text{pos}}^\top \quad \text{i} \times \text{fl}(\widehat{\mathbf{x}}_{\text{ang}} \widehat{\mathbf{x}}_{\text{pos}}^\top)^\top\right]^\top \in \mathbb{C}^{(M_1+1)(M_2+M_3)}, \tag{6.6}$$

$$\widehat{\mathbf{y}}_{\text{hyb}} = \frac{1}{\sqrt{2}}\left[\widehat{\mathbf{y}}_{\text{trig}}^\top \quad \text{fl}(\widehat{\mathbf{y}}_{\text{ang}} \widehat{\mathbf{y}}_{\text{trig}}^\top)^\top \quad \widehat{\mathbf{y}}_{\text{pos}}^\top \quad \text{i} \times \text{fl}(\widehat{\mathbf{y}}_{\text{ang}} \widehat{\mathbf{y}}_{\text{pos}}^\top)^\top\right]^\top \in \mathbb{C}^{(M_1+1)(M_2+M_3)} \tag{6.7}$$

where $\mathrm{fl}(\cdot)$ flattens the input matrix by concatenating its rows into a single vector. Then, $\mathrm{Re}\left(\widehat{\mathbf{x}}_{\mathrm{hyb}}^\top \widehat{\mathbf{y}}_{\mathrm{hyb}}\right)$ is exactly (6.4):

$$\begin{aligned}
&\mathrm{Re}\left(\widehat{\mathbf{x}}_{\mathrm{hyb}}^\top \widehat{\mathbf{y}}_{\mathrm{hyb}}\right) \\
&= \frac{1}{2}\mathrm{Re}\left(\widehat{\mathbf{x}}_{\mathrm{trig}}^\top \widehat{\mathbf{y}}_{\mathrm{trig}} + \mathrm{fl}(\widehat{\mathbf{x}}_{\mathrm{ang}}\widehat{\mathbf{x}}_{\mathrm{trig}}^\top)^\top \mathrm{fl}(\widehat{\mathbf{y}}_{\mathrm{ang}}\widehat{\mathbf{y}}_{\mathrm{trig}}^\top) + \widehat{\mathbf{x}}_{\mathrm{pos}}^\top \widehat{\mathbf{y}}_{\mathrm{pos}} - \mathrm{fl}(\widehat{\mathbf{x}}_{\mathrm{ang}}\widehat{\mathbf{x}}_{\mathrm{pos}}^\top)^\top \mathrm{fl}(\widehat{\mathbf{y}}_{\mathrm{ang}}\widehat{\mathbf{y}}_{\mathrm{pos}}^\top)\right) \\
&= \frac{1}{2}\left(\mathrm{Re}\left(\widehat{\mathbf{x}}_{\mathrm{trig}}^\top \widehat{\mathbf{y}}_{\mathrm{trig}}\right) + \mathrm{Re}\left(\mathrm{Tr}\left(\widehat{\mathbf{x}}_{\mathrm{ang}}\widehat{\mathbf{x}}_{\mathrm{trig}}^\top \widehat{\mathbf{y}}_{\mathrm{trig}}\widehat{\mathbf{y}}_{\mathrm{ang}}^\top\right)\right) + \widehat{\mathbf{x}}_{\mathrm{pos}}^\top \widehat{\mathbf{y}}_{\mathrm{pos}} - \mathrm{Tr}\left(\widehat{\mathbf{x}}_{\mathrm{ang}}\widehat{\mathbf{x}}_{\mathrm{pos}}^\top \widehat{\mathbf{y}}_{\mathrm{pos}}\widehat{\mathbf{y}}_{\mathrm{ang}}^\top\right)\right) \\
&= \frac{1}{2}\left(\mathrm{Re}\left(\widehat{\mathbf{x}}_{\mathrm{trig}}^\top \widehat{\mathbf{y}}_{\mathrm{trig}}\right) + \mathrm{Re}\left(\mathrm{Tr}\left(\widehat{\mathbf{y}}_{\mathrm{ang}}^\top \widehat{\mathbf{x}}_{\mathrm{ang}}\widehat{\mathbf{x}}_{\mathrm{trig}}^\top \widehat{\mathbf{y}}_{\mathrm{trig}}\right)\right) + \widehat{\mathbf{x}}_{\mathrm{pos}}^\top \widehat{\mathbf{y}}_{\mathrm{pos}} - \mathrm{Tr}\left(\widehat{\mathbf{y}}_{\mathrm{ang}}^\top \widehat{\mathbf{x}}_{\mathrm{ang}}\widehat{\mathbf{x}}_{\mathrm{pos}}^\top \widehat{\mathbf{y}}_{\mathrm{pos}}\right)\right) \\
&= \frac{1}{2}\left(\mathrm{Re}\left(\widehat{\mathbf{x}}_{\mathrm{trig}}^\top \widehat{\mathbf{y}}_{\mathrm{trig}}\right) + \left(\widehat{\mathbf{y}}_{\mathrm{ang}}^\top \widehat{\mathbf{x}}_{\mathrm{ang}}\right)\mathrm{Re}\left(\widehat{\mathbf{x}}_{\mathrm{trig}}^\top \widehat{\mathbf{y}}_{\mathrm{trig}}\right) + \widehat{\mathbf{x}}_{\mathrm{pos}}^\top \widehat{\mathbf{y}}_{\mathrm{pos}} - \left(\widehat{\mathbf{y}}_{\mathrm{ang}}^\top \widehat{\mathbf{x}}_{\mathrm{ang}}\right)\left(\widehat{\mathbf{x}}_{\mathrm{pos}}^\top \widehat{\mathbf{y}}_{\mathrm{pos}}\right)\right) \quad (6.8)
\end{aligned}$$

where we use $\mathrm{fl}(\mathbf{Z}_1)^\top \mathrm{fl}(\mathbf{Z}_2) = \mathrm{Tr}(\mathbf{Z}_1^\top \mathbf{Z}_2)$ and the permutation property of the matrix trace. (6.8) is clearly equivalent to (6.4). Hence, $\widehat{\mathbf{x}}_{\mathrm{hyb}}, \widehat{\mathbf{y}}_{\mathrm{hyb}}$ can be used in place of $\widehat{\mathbf{x}}, \widehat{\mathbf{y}}$ (2.5) in the random feature approximation of the Gaussian kernel matrix as described in Section 2.1.2.

The following result quantifies the variance of the hybrid estimator. Interestingly, we find that the variance is reduced by setting $\boldsymbol{\omega}^{(1,m)} = \boldsymbol{\omega}^{(2,m)}$ for all m instead of sampling $\{\boldsymbol{\omega}^{(1,m)}\}$ and $\{\boldsymbol{\omega}^{(2,m)}\}$ independently.

Theorem 12. *Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ with the angle θ between $\mathbf{x}, \mathbf{y}$. Let $M_1 = M_2$, $\widehat{\mathbf{x}}_{\mathrm{hyb}-1}, \widehat{\mathbf{y}}_{\mathrm{hyb}-1}$ be defined in the same way as (6.6,6.7) with sets $\{\boldsymbol{\omega}^{(1,m)}\}$ and $\{\boldsymbol{\omega}^{(2,m)}\}$ sampled independently from each other. Let $\widehat{\mathbf{x}}_{\mathrm{hyb}-2}, \widehat{\mathbf{y}}_{\mathrm{hyb}-2}$ be defined in the same way as in (6.6,6.7) with $\boldsymbol{\omega}^{(1,m)} = \boldsymbol{\omega}^{(2,m)}$ for all $1 \leq m \leq M_1$. Then:*

$$\begin{aligned}
\mathrm{Var}\,\widehat{\mathbf{x}}_{\mathrm{hyb}-1}^\top \widehat{\mathbf{y}}_{\mathrm{hyb}-1} &= \left(1 - \frac{\theta}{\pi}\right)^2 \mathrm{Var}\,\mathrm{Re}\left(\widehat{\mathbf{x}}_{\mathrm{trig}}^\top \widehat{\mathbf{y}}_{\mathrm{trig}}\right) + \frac{\theta^2}{\pi^2}\mathrm{Var}\,\widehat{\mathbf{x}}_{\mathrm{pos}}^\top \widehat{\mathbf{y}}_{\mathrm{pos}} \\
&\quad + \frac{\theta}{M_3\pi}\left(1 - \frac{\theta}{\pi}\right)\left(\mathrm{Var}\,\mathrm{Re}\left(\widehat{\mathbf{x}}_{\mathrm{trig}}^\top \widehat{\mathbf{y}}_{\mathrm{trig}}\right) + \mathrm{Var}\,\widehat{\mathbf{x}}_{\mathrm{pos}}^\top \widehat{\mathbf{y}}_{\mathrm{pos}}\right) \quad (6.9) \\
\mathrm{Var}\,\widehat{\mathbf{x}}_{\mathrm{hyb}-2}^\top \widehat{\mathbf{y}}_{\mathrm{hyb}-2} &= \mathrm{Var}\,\widehat{\mathbf{x}}_{\mathrm{hyb}-1}^\top \widehat{\mathbf{y}}_{\mathrm{hyb}-1} - \frac{2\theta}{M_1\pi}\left(1 - \frac{\theta}{\pi}\right)\left(1 - \frac{1}{M_3}\right) \\
&\quad \times K(\mathbf{x}, \mathbf{y})^2\left(1 - \cos(\|\mathbf{x}\|^2 - \|\mathbf{y}\|^2)\right) \leq \mathrm{Var}\,\widehat{\mathbf{x}}_{\mathrm{hyb}-1}^\top \widehat{\mathbf{y}}_{\mathrm{hyb}-1} \quad (6.10)
\end{aligned}$$

where $\mathrm{Var}\,\mathrm{Re}\left(\widehat{\mathbf{x}}_{\mathrm{trig}}^\top \widehat{\mathbf{y}}_{\mathrm{trig}}\right) = \frac{1}{2M_1}(1 - K(\mathbf{x}, \mathbf{y})^2)$ *and* $\mathrm{Var}\,\widehat{\mathbf{x}}_{\mathrm{pos}}^\top \widehat{\mathbf{y}}_{\mathrm{pos}} = \frac{1}{M_1}\left(\exp(4\mathbf{x}^\top \mathbf{y}) - K(\mathbf{x}, \mathbf{y})\right)$ *as follows from (2.10) and (4.4).*

Due to the results of this theorem, we opt for $M_1 = M_2$ and $\widehat{\mathbf{x}}_{\mathrm{hyb}-2}, \widehat{\mathbf{y}}_{\mathrm{hyb}-2}$ as the default choice of the hybrid estimator. The variance of this estimator compared to TrigRF and PosRF is illustrated in Figure 6.1. We observe that, indeed, the variance of HybRF estimator is zero

when $\theta \in \{0, \pi\}$, however that comes at a cost of not always outperforming PosRFs and TrigRFs in terms of variance.

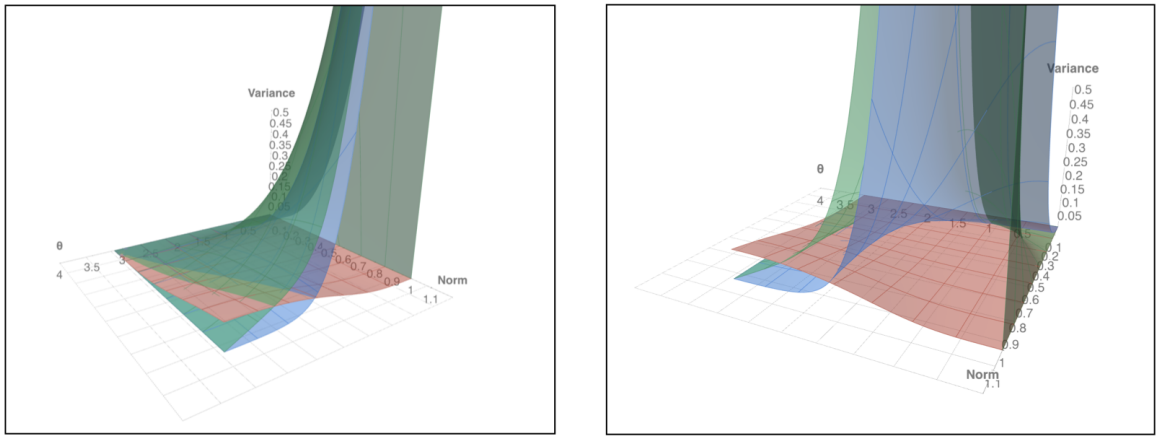


Fig. 6.1 Comparison of the variance of TrigRF (red), PosRF (blue) and HybRF (green) estimators expressed as functions of Norm $= \|\mathbf{x}\|^2 = \|\mathbf{y}\|^2$ and the angle θ between $\mathbf{x}$ and $\mathbf{y}$ (similar to Figure 4.1). We set $M = 4$, $M_1 = M_2 = M_3 = 1$ for a fair comparison since $M = (M_3 + 1)(M_1 + M_2)$.

6.3 Generalized exponential random features

In this section we generalize both trigonometric and positive RFs. After that we focus on one special case of this generalization which has an optimal variance in the set of positive-valued random features.

We will be looking for RFs of the following generalized exponential form for $\boldsymbol{\omega} \sim \mathcal{N}(0,1)^d$:

$$\begin{aligned} f^{(1)}_{\mathrm{GE}}(\boldsymbol{\omega}, \mathbf{x}) &= D\exp(A\|\boldsymbol{\omega}\|^2 + B\boldsymbol{\omega}^\top \mathbf{x} + C\|\mathbf{x}\|^2), \\ f^{(2)}_{\mathrm{GE}}(\boldsymbol{\omega}, \mathbf{y}) &= D\exp(A\|\boldsymbol{\omega}\|^2 + tB\boldsymbol{\omega}^\top \mathbf{y} + C\|\mathbf{y}\|^2), \end{aligned} \tag{6.11}$$

where $A, B, C, D \in \mathbb{C}$ and $t \in \{-1, +1\}$. It can be seen that $A = 0$, $B = \mathrm{i}$, $C = 0$, $D = 1$, $t = -1$ corresponds to trigonometric RFs and $A = 0$, $B = 1$, $C = -1$, $D = 1$, $t = 1$ corresponds to positive RFs. The next theorem describes the conditions under which $f^{(\cdot)}_{\mathrm{GE}}$ can be used to approximate the Gaussian kernel.

Theorem 13. *$p_{\mathrm{SG}}(\boldsymbol{\omega})$ and $f^{(\cdot)}_{\mathrm{GE}}$, defined in (6.11), satisfy Definition 1 if*

$$\mathrm{Re}\,(1 - 4A) > 0, \quad B = \sqrt{t(1-4A)}, \quad C = -(t+1)/2, \quad D = (\sqrt[4]{1-4A})^d. \tag{6.12}$$

Hence, A and t can be treated as free parameters and B, C, D as the dependent ones. We refer to the random features of type (6.11) satisfying (6.12) as *generalized exponential random features (GERFs)*. The variance of GERFs can be expressed through A and t as follows:

Theorem 14. *Let* $\mathrm{Re}\,(1-8A) > 0$. *The variance of Gaussian kernel estimator using GERFs for any* $\mathbf{x},\mathbf{y} \in \mathbb{R}^d$ *is given as*

$$\mathrm{Var}_{p_{\mathrm{SG}}(\boldsymbol{\omega})}\,\mathrm{Re}\left(f^{(1)}_{\mathrm{GE}}(\boldsymbol{\omega},\mathbf{x})f^{(2)}_{\mathrm{GE}}(\boldsymbol{\omega},\mathbf{y})\right) = \frac{1}{2}\exp\left(-(t+1)\left(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2\right)\right)$$
$$\times\left(\mathrm{Re}\left(\alpha_1\exp\left(\alpha_2\|\mathbf{x}+t\mathbf{y}\|^2\right)\right)+\alpha_3\exp\left(\alpha_4\|\mathbf{x}+t\mathbf{y}\|^2\right)\right)-K(\mathbf{x},\mathbf{y})^2. \tag{6.13}$$

where $\alpha_1 = \left(\sqrt{1+\frac{16A^2}{1-8A}}\right)^d$, $\alpha_2 = \left(t+\frac{t}{1-8A}\right)$, $\alpha_3 = \left(1+\frac{16|A|^2}{1-8\,\mathrm{Re}(A)}\right)^{d/2}$, $\alpha_4 = \left(\frac{t}{2}+\frac{t+2|1-4A|}{2(1-8\,\mathrm{Re}(A))}\right)$.

While it is unclear how to find a global minimum of the objective (6.13) with respect to $A \in \mathbb{C}$ (we leave this as an open question), $\mathrm{Re}\,(1-8A) > 0$ and $t \in \{-1,+1\}$, we observe that it's possible to find an optimum when we restrict A to be a real number and fix $t = +1$.

Theorem 15. *When* $t = +1$, *A is restricted to be a real number and* $\|\mathbf{x}+\mathbf{y}\|^2 > 0$, *the variance* (6.13) *is minimized when* $A = (1-1/\rho^*)/8$ *where* $0 < \rho^* < 1$,

$$\rho^* = \left(\sqrt{\left(2\|\mathbf{x}+\mathbf{y}\|^2+d\right)^2+8d\|\mathbf{x}+\mathbf{y}\|^2}-2\|\mathbf{x}+\mathbf{y}\|^2-d\right)/\left(4\|\mathbf{x}+\mathbf{y}\|^2\right). \tag{6.14}$$

Since $A = (1-1/\rho^*)/8$ and $0 < \rho^* < 1$, we conclude that $A < 0$. Thus, the corresponding estimator is bounded since the term $A\|\boldsymbol{\omega}\|^2$ prevails over the linear terms $B\boldsymbol{\omega}^\top\mathbf{x}$ and $tB\boldsymbol{\omega}^\top\mathbf{y}$ in (6.11).

From (6.12) it can be inferred that B,C,D are real when A is real and $t = +1$. Hence, $f^{(1)}_{\mathrm{GE}}(\boldsymbol{\omega},\mathbf{x})$, $f^{(2)}_{\mathrm{GE}}(\boldsymbol{\omega},\mathbf{y})$ are positive real numbers in this case. Furthermore, $t = +1, A = 0$ corresponds to PosRFs. Therefore, we refer to RFs with A defined according to (6.14) as *optimal positive random features* (OPRFs). Figure 6.2 illustrates the analytical variance reduction achieved via OPRFs.

In practice we are given with the sets $\{\mathbf{x}^{(i)}\}_{i=1}^{L}$, $\{\mathbf{y}^{(j)}\}_{j=1}^{L'}$ instead of a single pair $\mathbf{x},\mathbf{y}$ ($L = L'$ in Performer applications). For this reason, in (6.13,6.14) we can use the averages of $\|\mathbf{x}^{(i)}\|^2$, $\|\mathbf{y}^{(j)}\|^2$, $\|\mathbf{x}^{(i)}+t\mathbf{y}^{(j)}\|^2$ instead of $\|\mathbf{x}\|^2$, $\|\mathbf{y}\|^2$, $\|\mathbf{x}+t\mathbf{y}\|^2$. This heuristic is based on the assumption that all $\{\mathbf{x}^{(i)}\}$ and $\{\mathbf{y}^{(j)}\}$ are homogeneous and $\|\mathbf{x}^{(i)}\|^2$, $\|\mathbf{y}^{(j)}\|^2$, $\|\mathbf{x}^{(i)}+t\mathbf{y}^{(j)}\|^2$ are tightly concentrated around their mean. Computing the averages of $\|\mathbf{x}^{(i)}\|^2$, $\|\mathbf{y}^{(j)}\|^2$ takes $O(Ld)$ and $O(L'd)$ time respectively. Using the formula below, the average of $\|\mathbf{x}^{(i)}+t\mathbf{y}^{(j)}\|^2$

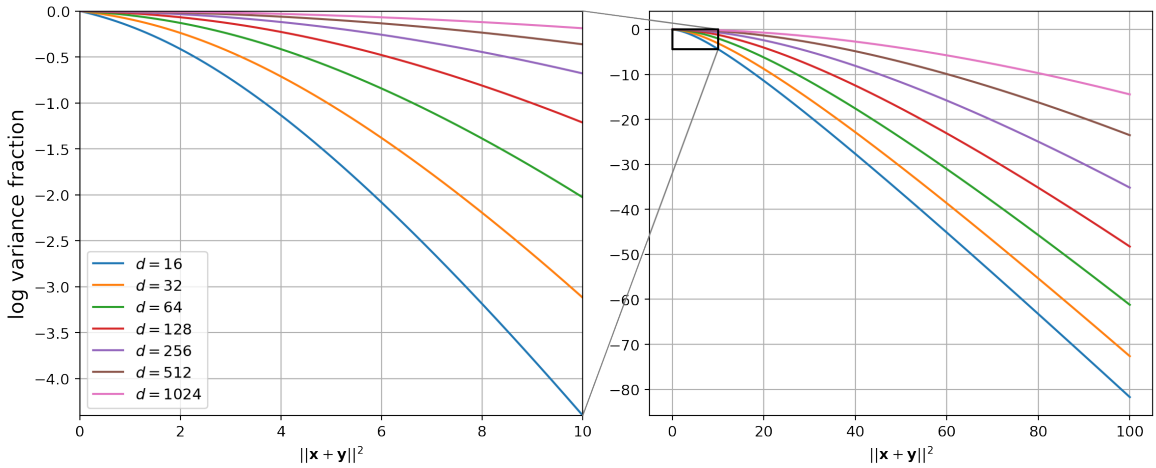


Fig. 6.2 The logarithm of the ratio of the variance of OPRF and PosRF mechanisms as a function of squared length of the sum of kernels' inputs $\|\mathbf{x}+\mathbf{y}\|^2$ (smaller values imply larger gains coming from OPRF). Different curves correspond to different dimensionalities. Based on the plots, OPRFs have $> e^{60}$ times smaller variance when $d = 64, \|\mathbf{x}+\mathbf{y}\|^2 = 100$.

can be computed with the same complexity:

$$\frac{1}{LL'}\sum_{i=1}^{L}\sum_{j=1}^{L'}\|\mathbf{x}^{(i)}+t\mathbf{y}^{(j)}\|^2 = \frac{1}{L}\sum_{i=1}^{L}\|\mathbf{x}^{(i)}\|^2 + \frac{2t}{LL'}\left(\sum_{i=1}^{L}\mathbf{x}^{(i)}\right)^{\top}\left(\sum_{j=1}^{L'}\mathbf{y}^{(j)}\right) + \frac{1}{L'}\sum_{j=1}^{L'}\|\mathbf{y}^{(j)}\|^2. \tag{6.15}$$

The closed-form solution for real A and $t = +1$ allows $O(1)$ time optimization of (6.13) after precomputing these statistics. In the general case of unbounded A and t we can rely on the numerical optimization of (6.13) with respect to $A \in \mathbb{C}$ and $t \in \{-1,+1\}$. Using these precomputed statistics each evaluation of (6.13) takes $O(1)$ time. As long as the total number of these evaluations is $O(LM(d+n))$ ($n = d+1$ for Transformers), it does not affect the total complexity.

All random feature mechanisms considered so far (TrigRFs, PosRFs, HybRFs, GERFs, OPRFs) are induced by the multivariate Gaussian distribution $\mathcal{N}(0,1)^d$. We next show that this condition is not required for the Gaussian kernel approximation. The next class of mechanisms comes from the stochastic approximation of the exponential's Taylor series and is induced by discrete distributions.

6.4 Discretely-induced random features

Take a discrete probabilistic distribution $p(\boldsymbol{\omega})$ where $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_d$ are i.i.d. with $\mathbb{P}(\boldsymbol{\omega}_l = k) = p_k$, $\sum_{k=0}^{\infty} p_k = 1$ and $p_k > 0$ for $k \in \{0\} \cup \mathbb{N}$. Note that, by Taylor series expansion of $\exp(\cdot)$,

$$K(\mathbf{x},\mathbf{y}) \exp\left(\frac{\|\mathbf{x}\|^2}{2}\right) \exp\left(\frac{\|\mathbf{y}\|^2}{2}\right) = \exp(\mathbf{x}^\top \mathbf{y}) = \prod_{l=1}^{d} \sum_{k=0}^{\infty} p_k \frac{\mathbf{x}_l^k \mathbf{y}_l^k}{p_k k!} = \mathbb{E}\left(\prod_{l=1}^{d} \mathscr{X}_l \prod_{l=1}^{d} \mathscr{Y}_l\right),$$

where $\mathscr{X}_l = \mathbf{x}_l^{\boldsymbol{\omega}_l} (\boldsymbol{\omega}_l!)^{-\frac{1}{2}} p_{\boldsymbol{\omega}_l}^{-\frac{1}{2}}, \mathscr{Y}_l = \mathbf{y}_l^{\boldsymbol{\omega}_l} (\boldsymbol{\omega}_l!)^{-\frac{1}{2}} p_{\boldsymbol{\omega}_l}^{-\frac{1}{2}}$. Thus we can define *discretely-induced random features (DIRFs)* providing Gaussian kernel estimation as follows:

$$f_{\mathrm{DI}}^{(1)}(\boldsymbol{\omega},\mathbf{x}) = f_{\mathrm{DI}}^{(2)}(\boldsymbol{\omega},\mathbf{x}) = f_{\mathrm{DI}}(\boldsymbol{\omega},\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}\|^2}{2}\right) \prod_{l=1}^{d} x_i^{\boldsymbol{\omega}_l} (\boldsymbol{\omega}_l!)^{-\frac{1}{2}} p_{\boldsymbol{\omega}_l}^{-\frac{1}{2}}. \tag{6.16}$$

Different instantiations of the above mechanism are given by different probabilistic distributions $\{p_k\}$. We will consider two prominent special cases: (a) Poisson, and (b) geometric distributions.

6.4.1 Poisson random features

If $\{p_k\}$ is a Poisson distribution, i.e. $p_k = e^{-\lambda}\lambda^k/k!$, $k \in \{0\} \cup \mathbb{N}$, then the corresponding RFs are defined as: $f_{\mathrm{pois}}^{(1)}(\boldsymbol{\omega},\mathbf{x}) = f_{\mathrm{pois}}^{(2)}(\boldsymbol{\omega},\mathbf{x}) = f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{x}) = \exp(\lambda d/2 - \|\mathbf{x}\|^2/2) \prod_{l=1}^{d} \mathbf{x}_l^{\boldsymbol{\omega}_l} \lambda^{-\boldsymbol{\omega}_l/2}$. Variance of these RFs, which we refer to as *Poisson random features (PoisRFs)*, has the following form:

Theorem 16. *For any $\mathbf{x},\mathbf{y} \in \mathbb{R}^d$, the variance of PoisRFs with the parameter $\lambda > 0$ is given by*

$$\mathrm{Var}_{p_{\mathrm{pois}}(\boldsymbol{\omega})}\left(f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{x}) f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{y})\right) = \exp\left(\lambda d + \lambda^{-1} \sum_{l=1}^{d} \mathbf{x}_l^2 \mathbf{y}_l^2 - \|\mathbf{x}\|^2 - \|\mathbf{y}\|^2\right) - K(\mathbf{x},\mathbf{y})^2. \tag{6.17}$$

The exp argument in (6.17) is convex as a function of $\lambda > 0$. By setting its derivative to zero, we find that $\lambda^* = d^{-1/2}(\sum_{l=1}^{d} \mathbf{x}_l^2 \mathbf{y}_l^2)^{1/2}$ gives the minimum of (6.17).

When, instead of a single pair $\mathbf{x},\mathbf{y}$, the sets $\{\mathbf{x}^{(i)}\}_{i=1}^{L}$, $\{\mathbf{y}^{(j)}\}_{j=1}^{L'}$ are provided, we can use the same homogeneity assumption as in Section 6.3 and substitute the average of $\sum_{l=1}^{d} (\mathbf{x}_l^{(i)})^2 (\mathbf{y}_l^{(j)})^2$ over $1 \le i \le L$, $1 \le j \le L'$ instead of $\sum_{l=1}^{d} \mathbf{x}_l^2 \mathbf{y}_l^2$. This average can be

computed efficiently in $O((L+L')d)$ time as follows:

$$\frac{1}{LL'}\sum_{i=1}^{L}\sum_{j=1}^{L'}\sum_{l=1}^{d}(\mathbf{x}_l^{(i)})^2(\mathbf{y}_l^{(j)})^2 = \frac{1}{LL'}\sum_{l=1}^{d}\left(\sum_{i=1}^{L}(\mathbf{x}_l^{(i)})^2\right)\left(\sum_{j=1}^{L}(\mathbf{y}_l^{(j)})^2\right). \tag{6.18}$$

After computing this statistic, we can calculate λ^* in $O(1)$ time using the analytic formula.

6.4.2 Geometric random features

If $\{p_k\}$ is a geometric distribution, i.e. $p_k = q(1-q)^k$, $k \in \{0\} \cup \mathbb{N}$, for a parameter $0 < q < 1$, then the corresponding RFs are defined as: $f^{(1)}_{\mathrm{geom}}(\boldsymbol{\omega}, \mathbf{x}) = f^{(2)}_{\mathrm{geom}}(\boldsymbol{\omega}, \mathbf{x}) = f_{\mathrm{geom}}(\boldsymbol{\omega}, \mathbf{x}) = q^{-d/2}e^{-\|\mathbf{x}\|^2/2}\prod_{l=1}^{d}\mathbf{x}_l^{\boldsymbol{\omega}_l}(1-q)^{-\boldsymbol{\omega}_l/2}(\boldsymbol{\omega}_l!)^{-1/2}$. We refer to such random features as *geometric random features (GeomRFs)*. Their variance has the following form:

Theorem 17. *For any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, the variance of GeomRFs with any $0 < q < 1$ is given as*

$$\mathrm{Var}_{p_{\mathrm{geom}}(\boldsymbol{\omega})}\left(f_{\mathrm{geom}}(\boldsymbol{\omega}, \mathbf{x}) f_{\mathrm{geom}}(\boldsymbol{\omega}, \mathbf{y})\right) = q^{-d}e^{-\|\mathbf{x}\|^2 - \|\mathbf{y}\|^2}\prod_{l=1}^{d} I_0(2(1-q)^{-\frac{1}{2}}|\mathbf{x}_l\mathbf{y}_l|) - K(\mathbf{x}, \mathbf{y})^2 \tag{6.19}$$

where $I_0(\cdot)$ is the modified Bessel function of the first kind of order 0.

Again as for the previously described mechanisms, when the sets $\{\mathbf{x}^{(i)}\}_{i=1}^{L}$, $\{\mathbf{y}^{(j)}\}_{j=1}^{L'}$ are given, we can use the averages of $|\mathbf{x}_l^{(i)}\mathbf{y}_l^{(j)}|$, $1 \leq l \leq d$, instead of $|\mathbf{x}_l\mathbf{y}_l|$ in (6.19) assuming a homogeneity of $\mathbf{x}^{(i)}$'s and $\mathbf{y}^{(j)}$'s. Each out of d averages can be computed in $O(L+L')$ time as follows:

$$\frac{1}{LL'}\sum_{i=1}^{L}\sum_{j=1}^{L'}|\mathbf{x}_l^{(i)}\mathbf{y}_l^{(j)}| = \frac{L}{L'}\left(\sum_{i=1}^{L}|\mathbf{x}_l^{(i)}|\right)\left(\sum_{j=1}^{L'}|\mathbf{y}_l^{(j)}|\right). \tag{6.20}$$

After the precomputation of these statistics, evaluation of (6.17) takes $O(d)$ time. A numerical optimization can be used to minimize (6.17) with respect to q. As long as the number of variance evaluations is $O((L+L')M(1+n/d))$, the total complexity estimate is not affected.

It's an open question whether the variance (6.19) has a closed form solution for the minimum. Furthermore, it's unclear whether there is a way to consider the general formulation of DIRFs and find a distribution $\{p_k\}$ minimizing the variance without relying on special cases of Poisson or geometric distribution. We leave these questions to future work.

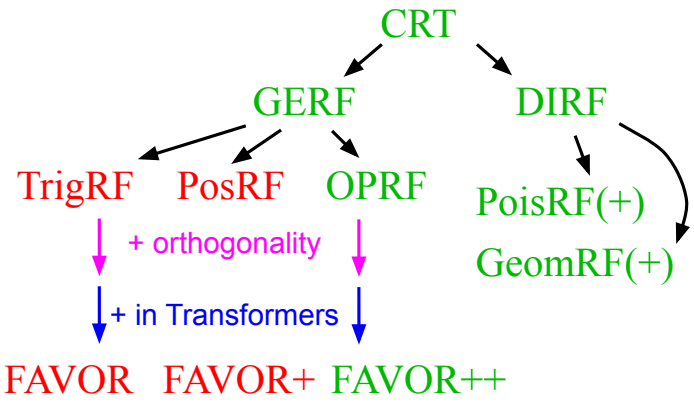


Fig. 6.3 A map of random feature methods for the Gaussian kernel approximation. Existing random features, random features proposed in this chapter. FAVOR++ will be defined later in Section 6.6.

6.4.3 Making discretely-induced random features positive

As can be inferred from (6.16), DIRFs are positive when all elements of $\mathbf{x}$ and $\mathbf{y}$ are positive. If this is not the case, one way to make them positive-valued is to take some vector $\mathbf{c} \in \mathbb{R}^d$ such that $\mathbf{c}_l < \mathbf{x}_l, \mathbf{y}_l$. An example of such a vector is given by $c_l = \min_i \min(\mathbf{x}_l^{(i)}, \mathbf{y}_l^{(i)}) - \varepsilon$ where $\varepsilon > 0$ is a small constant. Next, define $\widehat{\mathbf{x}}^{(i)} = \mathbf{x}^{(i)} - \mathbf{c}$, $\widehat{\mathbf{y}}^{(j)} = \mathbf{y}^{(j)} - \mathbf{c}$. Then, clearly, $\widehat{\mathbf{x}}^{(i)} - \widehat{\mathbf{y}}^{(j)} = \mathbf{x}^{(i)} - \mathbf{y}^{(j)}$, $K(\widehat{\mathbf{x}}^{(i)}, \widehat{\mathbf{y}}^{(j)}) = K(\mathbf{x}^{(i)}, \mathbf{y}^{(j)})$ and RFs can be used on $\widehat{\mathbf{x}}^{(i)}, \widehat{\mathbf{y}}^{(j)}$ which have positive entries. We refer to these variants of PoisRFs and GeomRFs as *PoisRF+* and *GeomRF+* respectively.

We refer to the union of GERFs and DIRFs with all their special cases as *chef's random tables (CRTs)*. Figure 6.3 illustrates all new and old types of random features as a diagram.

6.5 Concentration analysis

In this section, we perform a theoretical analysis of positive-valued GERFs including OPRFs and their orthogonal variants with respect to variance and concentration bounds. Interestingly, as in the case for TrigRFs and PosRFs, positive-valued GERFs including OPRFs benefit from taking block-orthogonal ensembles of projections $\boldsymbol{\omega}$ produced by Algorithm 2 instead of i.i.d. $\boldsymbol{\omega}$'s. We show below that block-orthogonal $\boldsymbol{\omega}$'s reduce the variance of GERFs with $A \in \mathbb{R}, t = 1$ which is the case for OPRF:

Theorem 18. *Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ and $\widehat{\mathbf{x}}_{\mathrm{GE}}, \widehat{\mathbf{y}}_{\mathrm{GE}}$ be defined according to (2.5) where $f^{(1)} = f^{(1)}_{\mathrm{GE}}, f^{(2)} = f^{(2)}_{\mathrm{GE}}$ with $A \in \mathbb{R}, t = 1$ in (6.11) and $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(M)}$ are i.i.d. samples from $\mathcal{N}(0,1)^d$. Further, let $\widehat{\mathbf{x}}_{\mathrm{ortGE}}, \widehat{\mathbf{y}}_{\mathrm{ortGE}}$ be defined according to (2.11) where $f^{(1)} = f^{(1)}_{\mathrm{GE}}, f^{(2)} =$*

$f^{(2)}_{\mathrm{GE}}$ *with the same* A, t *and* $\mathbf{\Omega}$ *is generated by Algorithm 2. If* $M \leq d$, *then*

$$\mathrm{Var}\widehat{\mathbf{x}}^{\top}_{\mathrm{ortGE}}\widehat{\mathbf{y}}_{\mathrm{ortGE}} \leq \mathrm{Var}\widehat{\mathbf{x}}^{\top}\widehat{\mathbf{y}} - \frac{2(M-1)}{M(d+2)}\left(K(\mathbf{x},\mathbf{y}) - \exp\left(-\frac{\|\mathbf{x}\|^2+\|\mathbf{y}\|^2}{2}\right)\right)^2.$$

If $M > d$ *and* M/d *is integer, then*

$$\mathrm{Var}\widehat{\mathbf{x}}^{\top}_{\mathrm{ort}}\widehat{\mathbf{y}}_{\mathrm{ort}} \leq \mathrm{Var}\widehat{\mathbf{x}}^{\top}\widehat{\mathbf{y}} - \frac{2(d-1)}{M(d+2)}\left(K(\mathbf{x},\mathbf{y}) - \exp\left(-\frac{\|\mathbf{x}\|^2+\|\mathbf{y}\|^2}{2}\right)\right)^2.$$

The analogous inequality can be obtained for TrigRFs only in the asymptotic sense for d large enough (Theorem 3). The theorem above is a special case of the theorem below which is an extension of Theorem 11 from Chapter 4. This extension supports the dependence of the function $\mathscr{G}$ on the norm $\|\boldsymbol{\omega}\|$. Again, a key property used in the proof is positivity of power series coefficients of $\mathscr{G}$.

Theorem 19. *Let* Ω *be an isotropic distribution on* $\mathbb{R}^d$ *and* $\widetilde{\Omega}$ *denote the distribution of* $\|\boldsymbol{\omega}\|$ *where* $\boldsymbol{\omega} \sim \Omega$. *Consider a function* $\mathscr{F} : \mathbb{R}^d \rightarrow \mathbb{R}$ *of the form:* $\mathscr{F}(\mathbf{z}) = \mathbb{E}\mathscr{G}(\mathbf{u}^{\top}\mathbf{z}, Y)$ *where* $\mathscr{G} : \mathbb{R} \times [0,+\infty) \rightarrow \mathbb{R}$, $\mathbf{u}$ *is sampled from* $\mathrm{Unif}(\mathbb{S}^{d-1})$ *and* Y *is sampled from* $\widetilde{\Omega}$ *distribution independently from* $\mathbf{u}$. *Assume furthermore that for every* $y \in [0,+\infty)$, *the function* $\mathscr{G}_y : \mathbb{R} \rightarrow \mathbb{R}$, *defined as* $\mathscr{G}_y(x) = \mathscr{G}(x,y)$, *satisfies:* $\mathscr{G}_y(x) = \sum_{k=0}^{\infty} a_k(y)x^k$ *for some* $a_0(y), a_1(y), \cdots \geq 0$. *Take two unbiased estimators of* $\mathscr{F}(\mathbf{z})$ *defined as follows for a natural* M:

$$\widehat{\mathscr{F}}^{\mathrm{iid}}_M = \frac{1}{M}\sum_{m=1}^{M}\mathscr{G}\left(\left(\frac{\boldsymbol{\omega}^{(\mathrm{iid},m)}}{\|\boldsymbol{\omega}^{(\mathrm{iid},m)}\|}\right)^{\top}\mathbf{z}, \|\boldsymbol{\omega}^{(\mathrm{iid},m)}\|\right),$$

$$\widehat{\mathscr{F}}^{\mathrm{ort}}_M = \frac{1}{M}\sum_{m=1}^{M}\mathscr{G}\left(\left(\frac{\boldsymbol{\omega}^{(\mathrm{ort},m)}}{\|\boldsymbol{\omega}^{(\mathrm{ort},m)}\|}\right)^{\top}\mathbf{z}, \|\boldsymbol{\omega}^{(\mathrm{ort},m)}\|\right)$$

where $\boldsymbol{\omega}^{(\mathrm{iid},1)}, \ldots, \boldsymbol{\omega}^{(\mathrm{iid},M)}$ *are i.i.d. samples from* Ω. $\boldsymbol{\omega}^{(\mathrm{ort},1)}, \ldots, \boldsymbol{\omega}^{(\mathrm{ort},M)}$ *are sampled in such a way that marginally* $\boldsymbol{\omega}^{(\mathrm{ort},m)} \sim \Omega$ *for all* $1 \leq m \leq M$ *but* $(\boldsymbol{\omega}^{(\mathrm{ort},m_1)})^{\top}\boldsymbol{\omega}^{(\mathrm{ort},m_2)} = 0$ *for all* $m_1 \neq m_2$, $\lfloor m_1/d \rfloor = \lfloor m_2/d \rfloor$. *Also, blocks of* d *consecutive* $\boldsymbol{\omega}^{(ort,m)}$*'s are mutually independent. If* $M \leq d$, *then*

$$\mathrm{Var}\,\widehat{\mathscr{F}}^{\mathrm{ort}}_M \leq \mathrm{Var}\,\widehat{\mathscr{F}}^{\mathrm{iid}}_M - \frac{2(M-1)}{M(d+2)}(\mathbb{E}\mathscr{F}(\mathbf{z}) - \mathbb{E}\mathscr{F}(\mathbf{0}_d))^2$$

where $Y \sim \widetilde{\Omega}$. *If* $M > d$ *and* M/d *is integer, then*

$$\operatorname{Var}\widehat{\mathscr{F}}_M^{\mathrm{ort}} \leq \operatorname{Var}\widehat{\mathscr{F}}_M^{\mathrm{iid}} - \frac{2(d-1)}{M(d+2)}(\mathbb{E}\mathscr{F}(\mathbf{z}) - \mathbb{E}\mathscr{F}(\mathbf{0}_d))^2.$$

We next show how to reduce Theorem 18 to Theorem 19. Define $\mathscr{G}(x,y) = D^2\exp(2Ay^2 + Bxy + 2C(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2))$ for $A,B,C,D \in \mathbb{R}$ satisfying Theorem 13 with $t = 1$. Take $\Omega = \mathscr{N}(0,1)^d$, $\widetilde{\Omega} = \chi(d)$, $\mathbf{z} = \mathbf{x} + \mathbf{y}$. Then, using the notation from Theorem 19,

$$\begin{aligned}\mathbb{E}\mathscr{F}(\mathbf{z}) &= \mathbb{E}D^2\exp(2AY^2 + BY\mathbf{u}^\top\mathbf{z} + C(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2)) \\ &= \mathbb{E}D^2\exp(2A\|\boldsymbol{\omega}\|^2 + B\boldsymbol{\omega}^\top(\mathbf{x}+\mathbf{y}) + C(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2)) \\ &= K(\mathbf{x},\mathbf{y})\end{aligned}$$

according to Theorem 13, where $\boldsymbol{\omega} = Y\mathbf{u} \sim \mathscr{N}(0,1)^d$. $\widehat{K}_M^{\mathrm{iid}}$, $\widehat{K}_M^{\mathrm{ort}}$ become the unbiased estimators of $K(\mathbf{x},\mathbf{y})$ applying M positive-valued GERFs with either i.i.d. $\boldsymbol{\omega}$'s or block-orthogonal $\boldsymbol{\omega}$'s. Since the power series of such $\mathscr{G}_y(x) = \mathscr{G}(x,y)$ has a form

$$\mathscr{G}_y(x) = \sum_{k=0}^{\infty} D^2\exp(2Ay^2 + C(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2))\frac{(By)^k}{k!}x^k,$$

the condition of non-negative power series coefficients is satisfied. Further, we have

$$\begin{aligned}&\mathbb{E}\mathscr{F}(\mathbf{0}_d) = \mathbb{E}D^2\exp(2A\|\boldsymbol{\omega}\|^2 + B\boldsymbol{\omega}^\top\mathbf{0}_d + C(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2)) \\ &\quad= \exp(C(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2)))\mathbb{E}D^2\exp(2A\|\boldsymbol{\omega}\|^2 + B\boldsymbol{\omega}^\top(\mathbf{0}_d + \mathbf{0}_d) + 2C(\|\mathbf{0}_d\|^2 + \|\mathbf{0}_d\|^2)) \\ &\quad= \exp(C(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2)))K(\mathbf{0}_d,\mathbf{0}_d) \\ &\quad= \exp(C(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2)) = \exp\left(-\frac{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2}{2}\right)\end{aligned}$$

where we use Theorem 13 for $\mathbf{x} = \mathbf{y} = \mathbf{0}_d$. Hence, we obtain the exact formulation of Theorem 18.

Our next contribution is strong concentration results for the positive GERF estimators with $A < 0$. OPRFs are a special case as discussed under Theorem 15. The condition $A < 0$ guarantees boundedness of the estimator and holds even when $\|\mathbf{x}+\mathbf{y}\|^2$ is substituted by the average (6.15) in (6.14) as long as (6.15) is greater than zero. These are the first results of this kind for positive-valued random features.

Theorem 20. *Let* $\mathbf{x},\mathbf{y} \in \mathbb{R}^d$ *and* $\widehat{\mathbf{x}}_{\mathrm{GE}}, \widehat{\mathbf{y}}_{\mathrm{GE}}$ *be defined according to (2.5) where* $f^{(1)} = f_{\mathrm{GE}}^{(1)}, f^{(2)} = f_{\mathrm{GE}}^{(2)}$ *with* $A \in \mathbb{R}, A < 0, t = 1$ *in (6.11) and* $\boldsymbol{\omega}^{(1)},\ldots,\boldsymbol{\omega}^{(M)}$ *are i.i.d. samples from* $\mathscr{N}(0,1)^d$. *Further, let* $\widehat{\mathbf{x}}_{\mathrm{ortGE}}, \widehat{\mathbf{y}}_{\mathrm{ortGE}}$ *be defined according to (2.11) where* $f^{(1)} =$

$f^{(1)}_{\mathrm{GE}}, f^{(2)} = f^{(2)}_{\mathrm{GE}}$ with the same A, t and $\mathbf{\Omega}$ is generated by Algorithm 2. The following is true for any $\varepsilon > 0$:

$$\mathbb{P}(|\widehat{\mathbf{x}}^{\top}_{\mathrm{GE}}\widehat{\mathbf{y}}_{\mathrm{GE}} - K(\mathbf{x},\mathbf{y})| \geq \varepsilon) \leq 2\exp\left(-\frac{M\varepsilon^2}{2}\exp\left(\frac{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2}{2A}\right)(1-4A)^{-\frac{d}{2}}\right).$$

Furthermore, for the orthogonal variant we have for all $\phi > 0$, $a > K(\mathbf{x},\mathbf{y})$:

$$\mathbb{P}\left((\widehat{\mathbf{x}}_{\mathrm{ortGE}})^{\top}\widehat{\mathbf{y}}_{\mathrm{ortGE}} > a\right) \leq \exp(-\phi M a)\left(\mathscr{M}_Z(\phi)^M\right)$$

where $\phi > 0$ and $Z = f^{(1)}_{\mathrm{GE}}(\boldsymbol{\omega},\mathbf{x})f^{(2)}_{\mathrm{GE}}(\boldsymbol{\omega},\mathbf{y})$, $\boldsymbol{\omega} \sim \mathscr{N}(0,1)^d$.

Finally, below we provide the result regarding the uniform concentration of self-attention approximation in Performers when using GERFs with $A < 0, t = 1$. We establish an upper bound on the minimal number of GERFs M needed for the uniform approximation of the self-attention matrix.

Theorem 21. *Assume that the L_2-norm of rows of matrices $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{L\times d}$ is upper-bounded by $R > 0$. Assume that $\widehat{\mathbf{A}}$ is an approximation of $\mathbf{A}$ from (2.16) obtained using M GERFs with $A < 0, t = 1$ with i.i.d. $\boldsymbol{\omega}^{(m)}$'s. Let $\varepsilon > 0$. The minimum M, which is enough for $\|\widehat{\mathbf{A}} - \mathbf{A}\|_{\infty} \leq \varepsilon$ with any constant probability, satisfies*

$$M = O\left(b_1\frac{d}{\varepsilon^2}\log\left(\frac{b_2 b_3}{\varepsilon}\right)\right), \quad b_1 = \exp\left(-\frac{3R^2}{\sqrt{d}A}\right),$$
$$b_2 = \sqrt{2}Rd^{-\frac{1}{4}}, \quad b_3 = 2(1-4A)^{\frac{d}{2}}\sqrt{b_1\left(\frac{R^2}{\sqrt{d}} + d^2\right)}.$$

The conclusion is similar to the result of Theorem 4 for TrigRF-based self-attention approximation: assuming that $d^{-1/2}R$ is constant, the number of random features grows as $O(d\log d)$ to reach the ε-level of approximation and doesn't depend on L. But this time random features are also positive-valued which gives stability during training.

6.6 Experiments

We present an extensive empirical evaluation of new random features proposed in this chapter. Additional details and results for each experiment can be found in Appendix 6.B.

6.6.1 Variance comparison

In this initial experiment, we sample synthetic pairs of vectors $\mathbf{x}, \mathbf{y}$ and evaluate variance of all random feature types based on the analytic formulae (2.10,4.4,6.10,6.13,6.17,6.19). Our goal is to check whether there are scenarios when the newly introduced random feature mechanisms have a smaller variance than the existing TrigRF and PosRF methods. We set $d = 64$ which is a standard query dimension in Performers. We use four different regimes for drawing $\mathbf{x}, \mathbf{y}$: `normal` corresponds to $\mathbf{x}, \mathbf{y}$ sampled from $\mathcal{N}(0, \gamma^2)^d$, `sphere` corresponds to $\mathbf{x}, \mathbf{y}$ sampled from Unif$(\mathbb{S}^{d-1})$ and multiplied by $\gamma > 0$, `heterogen` corresponds to $\mathbf{x}$ and $\mathbf{y}$ sampled from two heterogeneous distributions: $\mathcal{N}(0, \gamma^2)^d$ and $\mathcal{N}(\gamma, \gamma^2)^d$ and `mnist` corresponds to $\mathbf{x}, \mathbf{y}$ being random images from MNIST dataset (Deng, 2012) resized to 8×8 resolution, scaled by $\gamma > 0$ and flattened.

We use Brent method (Brent, 1971) for one dimensional optimization with 100 iterations for the minimization of q in GeomRF(+). We use two L-BFGS-B (Zhu et al., 1997) routines of 50 iterations each to minimize A in GERF for $t = -1$ and $+1$ and select the best performing one afterwards.

We sample 5 pairs of sets $\{\mathbf{x}^{(i)}\}_{i=1}^{L}$, $\{\mathbf{y}^{(j)}\}_{j=1}^{L}$, where $L = 1024$. On each pair of sets, we compute the variance of approximating $K(\mathbf{x}^{(i)}, \mathbf{y}^{(j)})$ for all pairs of $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(j)}$. Also, on each pair of sets of $\{\mathbf{x}^{(i)}\}_{1 \leq i \leq L}$, $\{\mathbf{y}^{(j)}\}_{1 \leq j \leq L}$, we compute statistics (6.15,6.18,6.20) and then use them to optimize parameters of the corresponding method. The means and standard deviations are reported for averaging over all pairs of $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(j)}$, over all 5 samples.

For a fair comparison, we count a complex-valued random feature as two real-valued features (real and imaginary channel) and assume $M = 4$ random features for complex-valued methods (TrigRFs, GERFs and HybRFs) and $M = 8$ random features for other, real-valued methods. For HybRFs, since $M = (M_3 + 1)(M_1 + M_2)$, we fix $M_1 = M_2 = M_3 = 1$.

The results are demonstrated in Figure 6.4. In many scenarios, new random feature variants outperform TrigRF and PosRF baselines. Among these improvements, GERF gives more than e^{80}, e^{125}, e^{10} times variance reduction compared to TrigRF in `normal`, `heterogen` and `mnist` respectively when $\gamma = 1$. OPRF and GeomRF+ give more than e^{75}, e^{125}, e^{7} times variance reduction compared to PosRF in `normal`, `heterogen` and `mnist` respectively when $\gamma = 1$. Overall, we observe that among the non-positive-valued random features, GERFs have the best performance. Among the positive-valued random features, GeomRFs+ and OPRFs have the best performance.

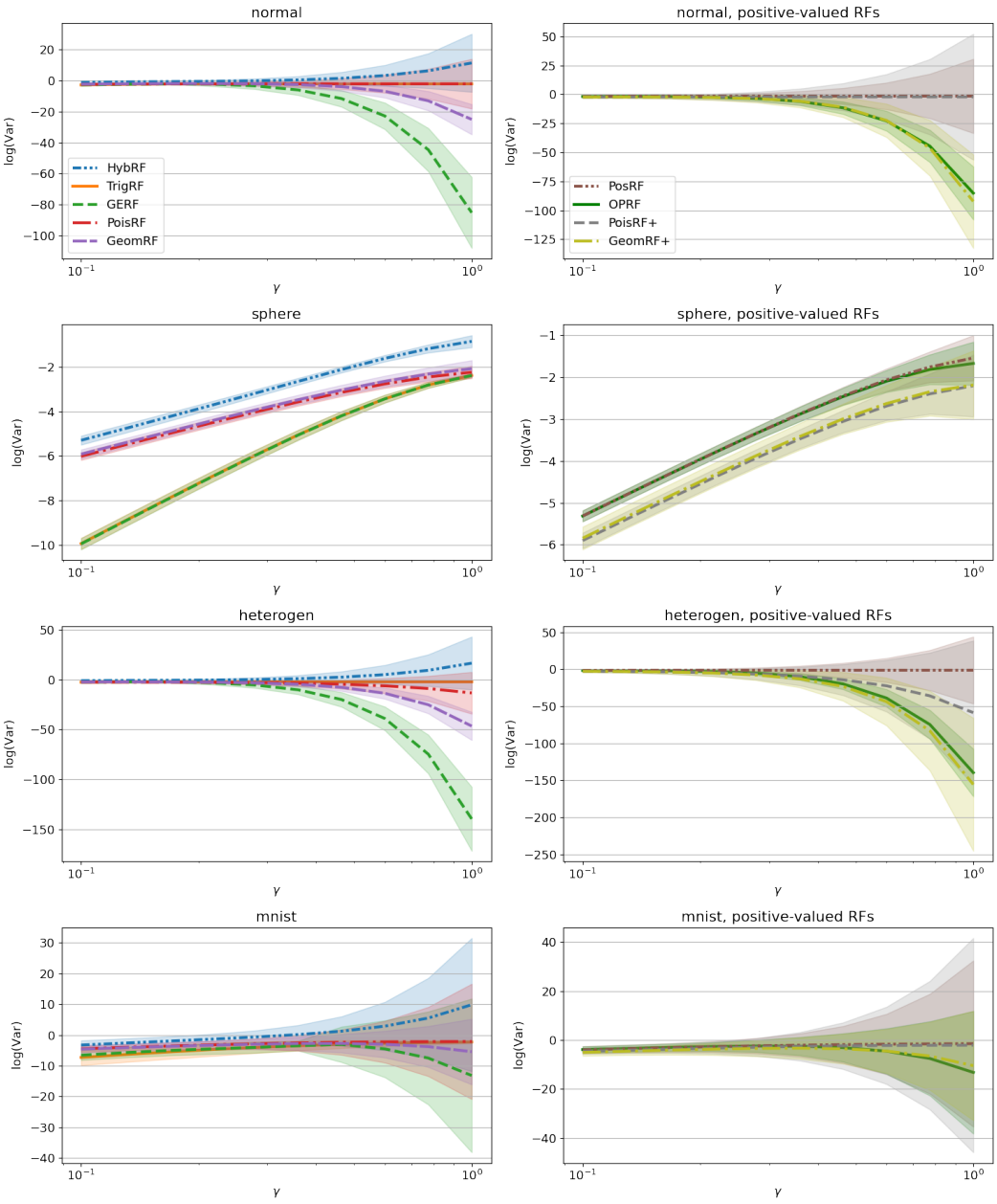


Fig. 6.4 Log-variance of different random feature mechanisms, mean and standard deviation. For each sampling method, we plot the results for non-positive and positive random features on separate plots for $0.1 \leq \gamma \leq 1$.

6.6.2 Non-parametric classification and FAVOR++

Our next experiment is a non-parametric classification where the probabilities are predicted by kernel regression with the Gaussian kernel as described in Section 2.1.1. Reusing notation from Section 2.1.1, $\mathbf{r}^{(i)}, \mathbf{r}^{*(j)} \in \mathbb{R}^n$ are one-hot encoded labels and n is the number of classes.

We tune the hyperparameter $\gamma > 0$ on a validation set. Using random feature approximation for the Gaussian kernel, we, with $O((d+n)LM)$ preprocessing, can predict each $\mathbf{r}^{*(j)}$ in $O((d+n)M)$ time per example instead of $O((d+n)L)$ for the exact computation, $M \ll L$.

Since the predicted class for the j'th test example is $\text{argmax}_{1\leq l\leq n}\mathbf{r}^{*(j)}$, we can ignore the denominator term in (2.1) and, therefore, use the non-positive-valued random features as well. The setup can be thought as evaluating self-attention over the whole dataset and in this sense it can be related to efficient Transformers. We opt for the classification benchmarks from the UCI Repository (Dua and Graff, 2017c) (Table 6.1). We randomly split each raw dataset into 90% for training, 5% for tuning γ and 5% for testing. These splits are fixed for all compared methods. γ is tuned on a log-uniform grid of 10 values from 10^{-2} to 10^{2}. For each γ and each method, we average accuracy for 50 seeds used to draw random features both during validation and testing (for the best γ only). As in the previous experiment, we use $M = 128$ for real-valued random features and $M = 64$ for complex-valued random features for a fair comparison. We evaluate the statistics (6.15,6.18,6.20) by taking $\mathbf{x}^{(i)}$'s and $\mathbf{y}^{(j)}$'s both from the train set, i.e. $L = L'$ is the train set size.

Table 6.1 UCI classification benchmarks used in the non-parametric classification.

Dataset	L
`abalone` (Nash and Tasmania., 1994)	3758
`banknote` (Dua and Graff, 2017a)	1233
`car` (Bohanec and Rajkovič, 1988)	1554
`yeast` (Horton and Nakai, 1996)	1334
`cmc` (Lim et al., 2000)	1324
`nursery` (Olave et al., 1989)	11664
`wifi` (Rohra et al., 2017)	1799
`chess` (Dua and Graff, 2017b)	25249

We use $\varepsilon = 10^{-8}$ when making input features positive in PoisRF+ and GeomRF+. $\mathbf{c}$ is inferred from the train set, and we clamp validation and test input features to be at least ε to guarantee that they are positive without leaking test data into $\mathbf{c}$. Numerical optimization of parameters in GERF, GeomRF(+) is performed in the same way as in the previous variance comparison experiment.

Table 6.2 compares i.i.d. and block-orthogonal variants of TrigRFs, PosRFs, GERFs and OPRFs. We observe that the block-orthogonal variant either doesn't harm, or improves the result in most cases. Further, two positive-valued random features (PosRF and OPRF) benefit from orthogonality when averaged over all benchmarks which is in line with our theoretical results showing that positive-valued random features are improved by orthogonality for all

values of d (Theorems 6, 19). Hence, we recommend the block-orthogonal variant as the default configuration for these methods.

Table 6.2 Non-parametric classification, comparison of i.i.d. variants / block-orthogonal variants. Test accuracy (%) is reported.

Dataset	TrigRF	PosRF	GERF	OPRF
abalone	12.0 / 12.0	15.5 / **16.0**	**17.7** / 17.0	16.7 / **17.1**
banknote	66.2 / 66.2	**83.9** / 83.4	93.2 / **92.4**	92.3 / **92.6**
car	66.3 / 66.3	68.9 / **69.2**	70.5 / **70.9**	**69.9** / 69.5
yeast	29.7 / 29.7	**34.6** / 34.4	42.8 / **42.9**	42.1 / **44.4**
cmc	46.6 / 46.6	44.7 / **45.1**	47.4 / **47.8**	**47.3** / 46.3
nursery	31.3 / 31.3	73.2 / **77.4**	63.8 / 63.8	75.8 / **78.9**
wifi	15.2 / 15.2	84.6 / **88.8**	93.0 / **93.3**	92.1 / **93.3**
chess	16.5 / 16.5	19.6 / **20.2**	20.4 / 20.4	**20.4** / 20.2
Average	35.5 / 35.5	53.1 / **54.3**	56.1 / 56.1	57.1 / **57.8**

Table 6.3 presents the final comparison of all methods. The best results are achieved by new random feature mechanisms, with GeomRF and OPRF performing particularly well. OPRF shows the best average performance, furthermore it is positive-valued meaning that it's compatible with stable Performer training. Therefore, our recommendation for practitioners is to opt for this method.

Due to OPRFs' impressive performance compared to PosRFs, we propose a new *FAVOR++* mechanism, in analogy to FAVOR and FAVOR++, which is a new superior method for low variance self-attention approximation. FAVOR++ is based on OPRFs with block-orthogonal random features when used in Transformer applications. In the remainder of the experimental section, we evaluate the new FAVOR++ mechanism in a number of Performer training setups in the areas of natural language processing, computer vision and speech modelling.

6.6.3 FAVOR++ in Performers

In this section, we evaluate FAVOR++ mechanism in various Performer training setups. We focus on comparing FAVOR++ to FAVOR+. Additional experimental details of each setup can be found in the appendix.

Natural language processing

In this setting, we test different Performer variants on the general language understanding evaluation (GLUE) benchmark (Wang et al., 2018), consisting of 8 different text

Table 6.3 Non-parametric classification, test accuracy (%) for all methods. The **best** result, second best.

Dataset	TrigRF	PosRF	HybRF	GERF	PoisRF	GeomRF	OPRF	PoisRF+	GeomRF+
abalone	12.0	16.0	12.2	17.0	18.0	**18.3**	17.1	14.0	15.1
banknote	66.2	83.4	66.2	92.4	84.4	**94.5**	92.6	80.1	85.6
car	66.3	69.2	66.3	**70.9**	66.3	66.3	69.5	66.3	67.2
yeast	29.7	34.4	29.7	42.9	36.9	35.9	**44.4**	29.7	31.0
cmc	46.6	45.1	46.6	**47.8**	46.6	47.3	46.3	35.5	43.5
nursery	31.3	77.4	31.4	63.8	77.1	77.1	**78.9**	77.3	71.0
wifi	15.2	88.8	28.2	93.3	95.3	**95.8**	93.3	77.2	82.9
chess	16.5	20.2	16.5	20.4	19.1	19.5	20.2	19.2	**22.5**
Average	35.5	54.3	37.14	56.1	55.5	56.8	**57.8**	49.9	52.3

classification tasks with the sequence length ranging from 32 to 128. We use the same training parameters as in (Devlin et al., 2018) and take a base Transformer configuration: $(h, s, d_{\text{ff}}, d_{\text{hid}}) = (12, 12, 3072, 784)$. We compare Performer with FAVOR+, Performer-ELU, Performer-ReLU, Performer with FAVOR++ and report the results in Table 6.4. We find that FAVOR++ outperforms all these low-rank Transformers in most GLUE tasks. In particular, FAVOR++ outperforms FAVOR+ on all GLUE tasks, demonstrating downstream effectiveness of variance reduction of the Gaussian kernel estimation. Moreover, warm starting with a checkpoint, which was pretrained via masked language modelling on a large text corpus (Devlin et al., 2018) (Uptrain FAVOR++), further improves performance. Note that the pretraining weights reused from (Devlin et al., 2018) were obtained using training with the exact self-attention (2.16), demonstrating that Performer with FAVOR++ is backward-compatible with the exact self-attention.

Table 6.4 GLUE results (development split) for Performer models. Number of training examples is reported below each task. MCC score is reported for CoLA, F1 score is reported for MRPC, Spearman correlation is reported for STS-B, and accuracy scores are reported for the other tasks (see (Wang et al., 2018) for the metric details). The **best** result, second best.

Variant	MNLI 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k
FAVOR+	80.26	89.53	87.13	90.58	53.17	85.07	83.82	67.59
ELU	80.72	90.05	89.09	91.51	48.43	86.68	85.05	**68.59**
ReLU	81.39	90.11	88.85	91.97	52.08	**87.64**	84.56	67.51
FAVOR++	81.25	90.15	89.58	92.00	54.95	85.62	85.78	67.87
Uptrain FAVOR++	**82.29**	**90.43**	**89.73**	**92.20**	**58.85**	85.90	**88.73**	67.63

Speech modelling

We compare FAVOR++ with FAVOR+ on masked language modelling where input strings are encoded recordings of human speech from the LibriSpeech ASR corpus (Panayotov et al., 2015) (sequence length $L = 512$). We use the 17-layer modification of the Transformer backbone from (Gulati et al., 2020) where we only modify multi-head self-attention blocks (2.15) by taking the efficient Performer self-attention instead of the exact self-attention. Each multi-head self-attention block consists of $h = 4$ self-attention heads. We use the word error rate (WER) metric – a standard way to evaluate speech models, see more details in (Gulati et al., 2020). We evaluate two variants with $M = 8$ and $M = 16$ (6.5). The WER improvement for FAVOR++ is substantial: 2.49% for $M = 8$ and 3.05% for $M = 16$ with a negligible $O(Ld) \ll O(LMd)$ overhead for computing (6.15) compared to FAVOR+.

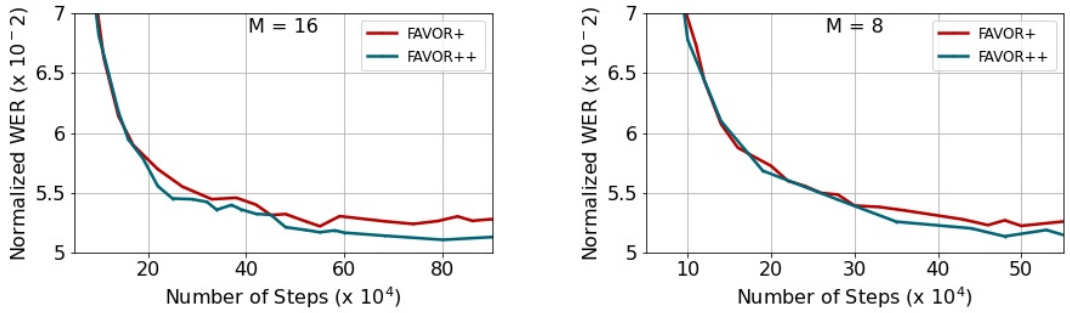


Fig. 6.5 Comparison of the masked speech modelling with FAVOR+ and FAVOR++ on the LibriSpeech (Panayotov et al., 2015) corpus for $M = 16$ and $M = 8$ RFs. We report a common word error rate (WER) metric.

Image recognition

To further showcase the need for more accurate Gaussian kernel approximation, we compare the performance of FAVOR+ and FAVOR++ self-attention approximation in ViT (Section 2.2.5) for image recognition on ImageNet dataset (Deng et al., 2009). We inject both mechanisms to the self-attention modules of ViT. In Figure 6.6 we show the results of training from scratch and uptraining from the pre-trained checkpoint taken from (He et al., 2021). Note that pretraining was performed for the exact self-attention (2.16). We see that, as opposed to FAVOR+, FAVOR++ is more stable and is able to improve performance especially for uptraining, demonstrating backward-compatibility with the exact self-attention. FAVOR+ didn't manage to adapt to the uptrained model and performed worse than when trained from scratch. We explain that by a big error of the FAVOR+ softmax estimator in this scenario.

Finally, we compare the computational complexity of FAVOR+ and FAVOR++. In Figure 6.6, the right plot shows the number of steps per second as a function of the sequence length L on the same hardware. We see that self-attention modules using FAVOR+ and FAVOR++ have a very similar computation time since the overhead of FAVOR++ is negligible. Moreover, for sequence lengths above 1000, training ViT with the exact self-attention becomes increasingly difficult due to out-of-memory errors.

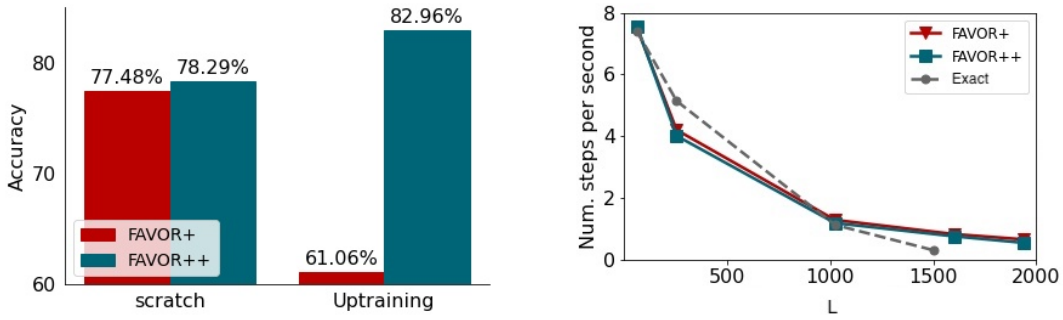


Fig. 6.6 Experimental results for image recognition. **(Left)** accuracy of training FAVOR+ and FAVOR++ on ImageNet from scratch and uptraining from pretrained weights. **(Right)** comparing the number of steps per second for different L for Performer with FAVOR+, FAVOR++ and the exact Transformer.

6.7 Discussion

In this chapter we introduced the whole zoo of new random features for the Gaussian kernel: hybrid random features (HybRFs) which unify TrigRFs and PosRFs in linear angle-based combinations and a family of non-hybrid variants which we refer to as chef's random tables (CRTs). The latter consists of two subfamilies. The first subfamily is generalized exponential random features (GERFs) which contain TrigRFs and PosRFs and a new optimal positive random features (OPRFs) as special cases. The second subfamily is discretely-induced random features (DIRFs) based on the unbiased approximation of exponent's Taylor series, with Poisson random features (PoisRFs) and geometric random features (GeomRFs) as special instantiations. OPRFs are obtained by optimizing variance of GERFs assuming that they are positive-valued (GERFs are complex-valued in the general case). Also, we can obtain positive-valued versions of PoisRFs and GeomRFs by a simple input-shifting trick – we refer to the resulting random features as PoisRFs+ and GeomRFs+. Since all examples of CRTs are parametrized, we proposed simple heuristics for optimizing their parameters by minimizing the variance where we substitute statistics evaluated over the sets $\{\mathbf{x}^{(i)}\}_{i=1}^{L}$

and $\{\mathbf{y}^{(j)}\}_{j=1}^{L'}$ into the variance expressions. Sometimes the variance can be minimized in a closed form (as in OPRFs, PoisRFs and PoisRFs+), in all other cases we use numerical optimization.

For the OPRF variant, we provided strong concentration results: strict variance improvement when using block-orthogonal random features compared to i.i.d. random features, exponentially fast concentration around the true kernel value and uniform concentration for the whole (unnormalized) self-attention matrix $\mathbf{A}$ when using OPRFs in Performers. Note that OPRFs are the first type of random features which is both positive-valued and bounded – a combination of two important feats missing in the earlier introduced TrigRFs and PosRFs.

We evaluated all newly proposed random features and compared them to the existing TrigRFs and PosRFs in a synthetic data sampling and non-parametric classification on UCI (Dua and Graff, 2017c). We find that OPRFs show a very competitive performance in both experiments. This, together with its fruitful theoretical properties, led to the emergence of FAVOR++ mechanism for self-attention approximation in Performers. FAVOR++ is based on the block-orthogonal version of OPRFs. We evaluated FAVOR++ in a number of large scale Performer training setups including general language understanding, speech modelling and image recognition. In all setups FAVOR++ outperforms or is competitive to other baselines. Furthermore, it always shows a better performance than FAVOR+ – previous iteration of efficient self-attention approximation mechanisms. We conclude that, in practice, FAVOR++ should be a default choice for the self-attention approximation mechanism in Performers.

Appendix 6.A Proofs

We will use the following lemma in several proofs:

Lemma 3. *Let $\alpha \in \mathbb{C}$, $\mathrm{Re}(\alpha) > 0$, and $\boldsymbol{\beta} \in \mathbb{C}^d$. Then, the following identity holds:*

$$\int_{\mathbb{R}^d} \exp\left(-\frac{\alpha}{2}\left[\boldsymbol{\omega}-\boldsymbol{\beta}\right]^2\right) \mathrm{d}\boldsymbol{\omega} = (2\pi)^{d/2}\left(\sqrt{\alpha}\right)^{-d}. \tag{6.21}$$

Proof. If both α and $\boldsymbol{\beta}$ are real, (6.21) is the well-known integral of the unnormalized multivariate Gaussian density with mean $\boldsymbol{\beta}$ and variance $\alpha^{-1}\mathbf{I}_d$. Since both the left and the right hand side in (6.21) are well-defined and analytic functions of α and β when $\mathrm{Re}(\alpha) > 0$, by the identity theorem (Walz, 2016) from the complex analysis we conclude that (6.21) holds when α and β are complex and $\mathrm{Re}(\alpha) > 0$. □

Proof of Theorem 12

Proof. Let $\square \in \{1,2\}$. Denote $\widehat{K}_{\text{hyb}-\square}(\mathbf{x},\mathbf{y}) = \widehat{\mathbf{x}}_{\text{hyb}-\square}^\top \widehat{\mathbf{y}}_{\text{hyb}-\square}$, $\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}) = \widehat{\mathbf{x}}_{\text{ang}}^\top \widehat{\mathbf{y}}_{\text{ang}}$, $\widehat{K}_{\text{pos}} = \widehat{\mathbf{x}}_{\text{pos}}^\top \widehat{\mathbf{y}}_{\text{pos}}$, $\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y}) = \text{Re}\left(\widehat{\mathbf{x}}_{\text{trig}}^\top \widehat{\mathbf{y}}_{\text{trig}}\right)$. Then, we have:

$$\begin{aligned}\text{Var}(\widehat{K}_{\text{hyb}-\square}(\mathbf{x},\mathbf{y})) = \frac{1}{4}\text{Var}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\right)\\ +\frac{1}{4}\text{Var}\left((1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right)\\ +\frac{1}{2}\text{Cov}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y}),(1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right)\end{aligned}$$

The following is also true:

$$\begin{aligned}\text{Var}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\right) &= \mathbb{E}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))^2\widehat{K}_{\text{exp}}(\mathbf{x},\mathbf{y})^2\right)\\ &\quad -\left(\mathbb{E}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\right)\right)^2\\ = \mathbb{E}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))^2\right)\mathbb{E}\left(\widehat{K}_{\text{exp}}(\mathbf{x},\mathbf{y})^2\right) &- \left(\mathbb{E}((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y})))\right)^2\left(\mathbb{E}(\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y}))\right)^2,\end{aligned}$$

where the last equality follows from the fact that $\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})$ and $\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y})$ are independent. Therefore, we have:

$$\text{Var}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\right) = \mathbb{E}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))^2\right)\mathbb{E}\left(\widehat{K}_{\text{exp}}(\mathbf{x},\mathbf{y})^2\right) - \frac{4\theta^2}{\pi^2}K(\mathbf{x},\mathbf{y})^2$$

Furthermore, since

$$\mathbb{E}(\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})^2) = \text{Var}\,\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})) + K(\mathbf{x},\mathbf{y})^2, \tag{6.22}$$

we obtain the following:

$$\begin{aligned}\text{Var}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\right) = \mathbb{E}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))^2\right)\left(\text{Var}\,\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y}) + K(\mathbf{x},\mathbf{y})^2\right)\\ -\frac{4\theta^2}{\pi^2}K(\mathbf{x},\mathbf{y})^2\end{aligned}$$

Let us now focus on the expression $\mathbb{E}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))^2\right)$. We have the following:

$$\mathbb{E}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))^2\right) = \mathbb{E}(1-2\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y})) + \widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))^2) = \frac{4\theta}{\pi} - 1 + \mathbb{E}(\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y})^2)$$

Denote $X_m = \text{sign}(\mathbf{x}^\top \boldsymbol{\omega}^{(3,m)})\text{sign}(\mathbf{y}^\top \boldsymbol{\omega}^{(3,m)})$ for $1 \leq m \leq M_3$. From the definition of the angle estimator, we get:

$$\mathbb{E}(\widehat{K}^2_{\text{ang}}(\mathbf{x},\mathbf{y})) = \mathbb{E}\left(\frac{1}{M_3^2}\left(\sum_{m=1}^{M_3} X_m\right)^2\right) = \frac{1}{M_3^2}\left(\sum_{m=1}^{M_3}\mathbb{E}(X_m^2) + 2\sum_{m_1<m_2}\mathbb{E}(X_{m_1}X_{m_2})\right) =$$
$$\frac{1}{M_3^2}\left(M_3 + 2\binom{M_3}{2}(\mathbb{E}(X_1))^2\right) = \frac{1}{M_3^2}\left(M_3 + M_3(M_3-1)\left(1-\frac{2\theta}{\pi}\right)^2\right)$$
$$= \frac{1}{M_3} + \left(1-\frac{1}{M_3}\right)\left(1-\frac{2\theta}{\pi}\right)^2.$$

Therefore, we conclude that

$$\mathbb{E}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))^2\right) = \frac{4\theta}{\pi} - 1 + \frac{1}{M_3} + \left(1-\frac{1}{M_3}\right)\left(1-\frac{2\theta}{\pi}\right)^2$$
$$= \frac{4\theta^2}{\pi^2} + \frac{4\theta}{\pi M_3}\left(1-\frac{\theta}{\pi}\right).$$

We conclude that

$$\text{Var}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\right) = \frac{4\theta}{\pi M_3}\left(1-\frac{\theta}{\pi}\right)K(\mathbf{x},\mathbf{y})^2$$
$$+\left(\frac{4\theta^2}{\pi^2} + \frac{4}{M_3}\frac{\theta}{\pi}\left(1-\frac{\theta}{\pi}\right)\right)\text{Var}\,\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y}).$$

Now we switch to the expression $\text{Var}\left((1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right)$. Using similar analysis as above, we get the following:

$$\text{Var}\left((1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right) = \mathbb{E}\left((1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))^2\right)\left(\text{Var}\,\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y}) + K(\mathbf{x},\mathbf{y})^2\right)$$
$$-4\left(1-\frac{\theta}{\pi}\right)^2 K(\mathbf{x},\mathbf{y})^2.$$

This time we need to compute the expression: $\mathbb{E}\left((1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))^2\right)$. We have the following:

$$\mathbb{E}\left((1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))^2\right) = \mathbb{E}(1+2\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}) + \widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y})^2)$$
$$= 1 + 2\left(1-\frac{2\theta}{\pi}\right) + \frac{1}{M_3} + \left(1-\frac{1}{M_3}\right)\left(1-\frac{2\theta}{\pi}\right)^2 = 4\left(\left(1-\frac{\theta}{\pi}\right)^2 + \frac{\theta}{\pi M_3}\left(1-\frac{\theta}{\pi}\right)\right)$$

where we used already derived formulae for $\mathbb{E}(\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y})^2)$. We conclude that:

$$\begin{aligned}\operatorname{Var}\left((1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right) &= \frac{4\theta}{\pi M_3}\left(1-\frac{\theta}{\pi}\right)K(\mathbf{x},\mathbf{y})^2 \\ &+\left(4\left(1-\frac{\theta}{\pi}\right)^2+\frac{4\theta}{\pi M_3}\left(1-\frac{\theta}{\pi}\right)\right)\operatorname{Var}\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y}).\end{aligned}$$

From the above, we obtain:

$$\begin{aligned}\operatorname{Var}\widehat{K}_{\text{hyb}-\square}(\mathbf{x},\mathbf{y}) &= \frac{2\theta}{\pi M_3}\left(1-\frac{\theta}{\pi}\right)K(\mathbf{x},\mathbf{y})^2+\left(\frac{\theta^2}{\pi^2}+\frac{\theta}{\pi M_3}\left(1-\frac{\theta}{\pi}\right)\right)\operatorname{Var}\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y}) \\ &+\left(\left(1-\frac{\theta}{\pi}\right)^2+\frac{\theta}{\pi M_3}\left(1-\frac{\theta}{\pi}\right)\right)\operatorname{Var}\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y}) \\ &+\frac{1}{2}\operatorname{Cov}((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y}),(1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})).\end{aligned}$$

Thus it remains to compute $\operatorname{Cov}((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y}),(1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y}))$. We have:

$$\begin{aligned}&\operatorname{Cov}((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y}),(1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})) \\ &= \mathbb{E}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y})^2)\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right) \\ &-\mathbb{E}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\right)\mathbb{E}\left((1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right) \\ &= \mathbb{E}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y})^2)\right)\mathbb{E}\left(\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right) \\ &-\mathbb{E}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\right)\mathbb{E}\left(\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\right)\mathbb{E}\left((1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\right)\mathbb{E}\left(\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right)\end{aligned}$$

where the last equation follows from the fact that $\{\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y}),\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\}$ and $\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y})$ are independent. Thus we deduce:

$$\begin{aligned}&\operatorname{Cov}((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y}),(1+\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y}))\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})) \\ &= \mathbb{E}\left((1-\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y})^2)\right)\mathbb{E}\left(\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right)-\frac{2\theta}{\pi}\left(2-\frac{2\theta}{\pi}\right)K(\mathbf{x},\mathbf{y})^2 \\ &= \left(1-\mathbb{E}(\widehat{K}_{\text{ang}}(\mathbf{x},\mathbf{y})^2)\right)\mathbb{E}\left(\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right)-\frac{4\theta}{\pi}\left(1-\frac{\theta}{\pi}\right)K(\mathbf{x},\mathbf{y})^2 \\ &= \frac{4\theta}{\pi}\left(1-\frac{\theta}{\pi}\right)\left(\left(1-\frac{1}{M_3}\right)\mathbb{E}\left(\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right)-K(\mathbf{x},\mathbf{y})^2\right),\end{aligned}$$

where we again used already derived formulae for $\mathbb{E}(\widehat{K}_{\mathrm{ang}}(\mathbf{x},\mathbf{y})^2)$. Therefore, we conclude that:

$$\begin{aligned}\mathrm{Var}(\widehat{K}_{\mathrm{hyb}-\square}(\mathbf{x},\mathbf{y})) = \frac{\theta}{\pi}\left(1-\frac{\theta}{\pi}\right)\left(\frac{\mathrm{Var}\,\widehat{K}_{\mathrm{pos}}(\mathbf{x},\mathbf{y})+\mathrm{Var}\,\widehat{K}_{\mathrm{trig}}(\mathbf{x},\mathbf{y})}{M_3}\right.\\ +2\left(1-\frac{1}{M_3}\right)\left(\mathbb{E}\left(\widehat{K}_{\mathrm{pos}}(\mathbf{x},\mathbf{y})\widehat{K}_{\mathrm{trig}}(\mathbf{x},\mathbf{y})\right)-K(\mathbf{x},\mathbf{y})^2\right)\Bigg)\\ +\frac{\theta^2}{\pi^2}\mathrm{Var}\,\widehat{K}_{\mathrm{pos}}(\mathbf{x},\mathbf{y})+\left(1-\frac{\theta}{\pi}\right)^2\mathrm{Var}\,\widehat{K}_{\mathrm{trig}}(\mathbf{x},\mathbf{y}). \qquad (6.23)\end{aligned}$$

Now, if $\square = 1$, we have

$$\mathbb{E}\left(\widehat{K}_{\mathrm{pos}}(\mathbf{x},\mathbf{y})\widehat{K}_{\mathrm{trig}}(\mathbf{x},\mathbf{y})\right) = \mathbb{E}\widehat{K}_{\mathrm{pos}}(\mathbf{x},\mathbf{y})\mathbb{E}\widehat{K}_{\mathrm{trig}}(\mathbf{x},\mathbf{y}) = K(\mathbf{x},\mathbf{y})^2$$

due to the independence of $\widehat{K}_{\mathrm{trig}}(\mathbf{x},\mathbf{y})$ and $\widehat{K}_{\mathrm{pos}}(\mathbf{x},\mathbf{y})$, resulting in (6.9). Next, assume that $\square = 2$, i.e. $\boldsymbol{\omega}^{(1,m)} = \boldsymbol{\omega}^{(2,m)}$ for $1 \leq m \leq M_1$. It remains to compute $\mathbb{E}\left(\widehat{K}_{\mathrm{pos}}(\mathbf{x},\mathbf{y})\widehat{K}_{\mathrm{trig}}(\mathbf{x},\mathbf{y})\right)$. Denote:

$$Y_m = f^{(1)}_{\mathrm{pos}}(\boldsymbol{\omega}^{(1,m)},\mathbf{x})f^{(2)}_{\mathrm{pos}}(\boldsymbol{\omega}^{(1,m)},\mathbf{y}) = \exp((\boldsymbol{\omega}^{(1,m)})^\top(\mathbf{x}+\mathbf{y}) - \|\mathbf{x}\|^2 - \|\mathbf{y}\|^2)$$

for $1 \leq m \leq M_1$ and

$$Z_m = \mathrm{Re}\left(f^{(1)}_{\mathrm{trig}}(\boldsymbol{\omega}^{(1,m)},\mathbf{x})f^{(2)}_{\mathrm{trig}}(\boldsymbol{\omega}^{(1,m)},\mathbf{y})\right) = \mathrm{Re}\left(\exp(\mathrm{i}(\boldsymbol{\omega}^{(1,m)})^\top(\mathbf{x}-\mathbf{y}))\right)$$

for $1 \leq m \leq M_1$. Then

$$\begin{aligned}\mathbb{E}\left(\widehat{K}_{\mathrm{pos}}(\mathbf{x},\mathbf{y})\widehat{K}_{\mathrm{trig}}(\mathbf{x},\mathbf{y})\right) &= \mathbb{E}\left(\frac{\sum_{m=1}^{M_1}Y_m\sum_{m=1}^{M_1}Z_m}{M_1^2}\right) = \frac{1}{M_1^2}\mathbb{E}\left(\sum_{m=1}^{M_1}Y_mZ_m + \sum_{m_1\neq m_2}Y_{m_1}Z_{m_2}\right)\\ &= \frac{1}{M_1}\mathbb{E}(Y_1Z_1) + \frac{2}{M_1^2}\binom{M_1}{2}\mathbb{E}(Y_1)\mathbb{E}(Z_2)\end{aligned}$$

where we used the fact that different Y_m have the same distributions, different Z_m have the same distributions, and furthermore for $m_1 \neq m_2$: $\widehat{Y}_{m_1}$ and $\widehat{Z}_{m_2}$ are independent since the corresponding $\boldsymbol{\omega}^{(1,m_1)}$ and $\boldsymbol{\omega}^{(1,m_2)}$ are chosen independently. We have $\mathbb{E}(Y_1) = \mathbb{E}(Z_2) = K(\mathbf{x},\mathbf{y})$ and

$$\begin{aligned}\mathbb{E}(Y_1Z_1) &= \mathbb{E}\left(\exp(\boldsymbol{\omega}^\top(\mathbf{x}+\mathbf{y}) - \|\mathbf{x}\|^2 - \|\mathbf{y}\|^2)\,\mathrm{Re}\left(\exp(\mathrm{i}\boldsymbol{\omega}^\top(\mathbf{x}-\mathbf{y}))\right)\right)\\ &= \mathrm{Re}\left(\mathbb{E}\left(\exp(\boldsymbol{\omega}^\top(\mathbf{x}+\mathbf{y}+\mathrm{i}(\mathbf{x}-\mathbf{y})))\right)\right)\exp(-\|\mathbf{x}\|^2-\|\mathbf{y}\|^2),\end{aligned}$$

where we denote $\boldsymbol{\omega} = \boldsymbol{\omega}^{(1,1)}$. We conclude that

$$\mathbb{E}\left(\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right) = \frac{1}{M_1}\text{Re}\left(\mathbb{E}\left(\exp(\boldsymbol{\omega}^\top(\mathbf{x}+\mathbf{y}+\text{i}(\mathbf{x}-\mathbf{y})))\right)\right) \times \exp(-\|\mathbf{x}\|^2 - \|\mathbf{y}\|^2) + \left(1-\frac{1}{M_1}\right)K(\mathbf{x},\mathbf{y})^2. \tag{6.24}$$

We further deduce that

$$\begin{aligned}
\mathbb{E}\left(\exp(\boldsymbol{\omega}^\top(\mathbf{x}+\mathbf{y}-\text{i}(\mathbf{x}-\mathbf{y})))\right) &= (2\pi)^{-d/2}\int \exp\left(-\|\boldsymbol{\omega}\|^2/2 + \boldsymbol{\omega}^\top(\mathbf{x}+\mathbf{y}+\text{i}(\mathbf{x}-\mathbf{y}))\right)\text{d}\boldsymbol{\omega} \\
&= \int \exp\left(-[\boldsymbol{\omega}-(\mathbf{x}+\mathbf{y}+\text{i}(\mathbf{x}-\mathbf{y}))]^2/2 + [\mathbf{x}+\mathbf{y}+\text{i}(\mathbf{x}-\mathbf{y})]^2/2\right)\text{d}\boldsymbol{\omega} \\
&= \exp\left([\mathbf{x}+\mathbf{y}+\text{i}(\mathbf{x}-\mathbf{y})]^2/2\right)\int \exp\left(-[\boldsymbol{\omega}-(\mathbf{x}+\mathbf{y}+\text{i}(\mathbf{x}-\mathbf{y}))]^2/2\right)\text{d}\boldsymbol{\omega} \\
&= \exp\left([\mathbf{x}+\mathbf{y}+\text{i}(\mathbf{x}-\mathbf{y})]^2/2\right) \\
&= \exp\left(2\mathbf{x}^\top\mathbf{y} + \text{i}(\|\mathbf{x}\|^2 - \|\mathbf{y}\|^2)\right)
\end{aligned}$$

where we use Lemma 3 with $\alpha = 1$, $\boldsymbol{\beta} = \mathbf{x}+\mathbf{y}+\text{i}(\mathbf{x}-\mathbf{y})$. By substituting this result into (6.24), we get

$$\begin{aligned}
\mathbb{E}\left(\widehat{K}_{\text{pos}}(\mathbf{x},\mathbf{y})\widehat{K}_{\text{trig}}(\mathbf{x},\mathbf{y})\right) &= \frac{1}{M_1}\text{Re}\left(\exp\left(2\mathbf{x}^\top\mathbf{y} - \|\mathbf{x}\|^2 - \|\mathbf{y}\|^2 + \text{i}(\|\mathbf{x}\|^2 - \|\mathbf{y}\|^2)\right)\right) \\
&+ \left(1-\frac{1}{M_1}\right)K(\mathbf{x},\mathbf{y})^2 = \left(1-\frac{1}{M_1}(1-\cos(\|\mathbf{x}\|^2 - \|\mathbf{y}\|^2))\right)K(\mathbf{x},\mathbf{y})^2 \leq K(\mathbf{x},\mathbf{y})^2
\end{aligned}$$

since $\cos(z) \leq 1$ for any z, resulting in (6.10).

□

Proof of Theorem 13

Proof. We rewrite (2.3) for $f_{\text{GE}}^{(\cdot)}$ and deduce that

$$\mathbb{E}\left(f_{\text{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\text{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})\right) = (2\pi)^{-d/2}D^2\int_{\mathbb{R}^d}\exp(-\|\boldsymbol{\omega}\|^2/2 + 2A\|\boldsymbol{\omega}\|^2 + B\boldsymbol{\omega}^\top(\mathbf{x}+t\mathbf{y}) + C(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2))\text{d}\boldsymbol{\omega} \tag{6.25}$$

where we express expectation as an integral and use definitions of $f_{\text{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})$, $f_{\text{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})$ and $p_{\text{SG}}(\boldsymbol{\omega})$. Next, we move out constant terms from the integral and put $\boldsymbol{\omega}$ into an elementwise

square of difference:

$$\int_{\mathbb{R}^d} \exp(-\|\boldsymbol{\omega}\|^2/2 + 2A\|\boldsymbol{\omega}\|^2 + B\boldsymbol{\omega}^\top(\mathbf{x}+t\mathbf{y}) + C(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2))\mathrm{d}\boldsymbol{\omega}$$
$$= \exp\left(\frac{B^2}{2(1-4A)}\|\mathbf{x}+t\mathbf{y}\|^2 + C(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2)\right)$$
$$\times \int_{\mathbb{R}^d} \exp\left(-\frac{1}{2}(1-4A)\left[\boldsymbol{\omega} - \frac{B}{1-4A}(\mathbf{x}+t\mathbf{y})\right]^2\right)\mathrm{d}\boldsymbol{\omega} \tag{6.26}$$

Next, we use Lemma 3 with $\alpha = 1-4A$ and $\boldsymbol{\beta} = (B/(1-4A))(\mathbf{x}+t\mathbf{y})$:

$$\int_{\mathbb{R}^d} \exp\left(-\frac{1}{2}(1-4A)\left[\boldsymbol{\omega} - \frac{B}{1-4A}(\mathbf{x}+t\mathbf{y})\right]^2\right)\mathrm{d}\boldsymbol{\omega} = (2\pi)^{d/2}\left(\sqrt{1-4A}\right)^{-d}. \tag{6.27}$$

Combining (6.25, 6.26, 6.27) together, we conclude that

$$\mathbb{E}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})\right) = D^2\left(\sqrt{1-4A}\right)^{-d}$$
$$\times \exp\left(\frac{B^2}{2(1-4A)}\|\mathbf{x}+t\mathbf{y}\|^2 + C(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2)\right). \tag{6.28}$$

The right hand side of (6.28) is $K(\mathbf{x},\mathbf{y})$ if the following conditions are satisfied in addition to $\mathrm{Re}\,(1-4A) > 0$:

$$D^2 = (\sqrt{1-4A})^d, \quad tB^2 = (1-4A), \quad \frac{B^2}{2(1-4A)} + C = -\frac{1}{2}. \tag{6.29}$$

(6.29) is satisfied when (6.12) takes place. The final observation is that $\mathrm{Re}\,(\cdot)$ is a linear operation and therefore, if (6.29) is satisfied,

$$\mathbb{E}\,\mathrm{Re}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})\right) = \mathrm{Re}\left(\mathbb{E}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})\right)\right) = \mathrm{Re}\,(K(\mathbf{x},\mathbf{y})) = K(\mathbf{x},\mathbf{y}).$$

□

It's possible to use other complex roots in (6.12) rather than just principal roots. However, in the proof of Theorem 14, we will only use (6.29) and, therefore, the variance is the same when other complex roots are used. We opt for principal roots for simplicity.

Proof of Theorem 14

Proof. We first use $\operatorname{Var} Z = \mathbb{E}Z^2 - (\mathbb{E}Z)^2$ which holds for any random variable Z, e.g. $\operatorname{Re}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})\right)$:

$$\operatorname{Var}_{p_{\mathrm{SG}}(\boldsymbol{\omega})} \operatorname{Re}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})\right) = \mathbb{E}\operatorname{Re}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})\right)^2 - \left(\mathbb{E}\operatorname{Re}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})\right)\right)^2 \tag{6.30}$$

The second term transforms into $\left(\mathbb{E}\operatorname{Re}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})\right)\right)^2 = K(\mathbf{x},\mathbf{y})^2$. As for the first term, we use $\operatorname{Re}(z) = \frac{1}{2}(z+\bar{z})$ and get:

$$\mathbb{E}\operatorname{Re}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})\right)^2 = \frac{1}{4}\mathbb{E}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y}) + \overline{f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})}\right)^2 \tag{6.31}$$

We unfold the square of the sum:

$$\mathbb{E}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y}) + \overline{f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})}\right)^2 = \mathbb{E}\Big(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})^2 f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})^2 + 2|f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})|^2|f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})|^2 + \overline{f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})^2 f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})^2}\Big) \tag{6.32}$$

Further, we observe that, again:

$$f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})^2 f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})^2 + \overline{f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})^2 f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})^2} = 2\operatorname{Re}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})^2 f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})^2\right). \tag{6.33}$$

We use that in (6.32) and also put expectation inside the sum and $\operatorname{Re}(\cdot)$ due to linearity:

$$\begin{aligned}&\mathbb{E}\left(2\operatorname{Re}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})^2 f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})^2\right) + 2|f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})|^2|f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})|^2\right)\\ &= 2\operatorname{Re}\left(\mathbb{E}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})^2 f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})^2\right)\right) + 2\mathbb{E}\left(|f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})|^2|f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})|^2\right)\end{aligned} \tag{6.34}$$

Denote $f_{\mathrm{GE}}^{(1)}$ and $f_{\mathrm{GE}}^{(2)}$ with parameters A,B,C,D as $f_{A,B,C,D}^{(1)}, f_{A,B,C,D}^{(2)}$. Then according to (6.11),

$$f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})^2 = f_{2A,2B,2C,D^2}^{(1)}(\boldsymbol{\omega},\mathbf{x}), \quad f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{x})^2 = f_{2A,2B,2C,D^2}^{(2)}(\boldsymbol{\omega},\mathbf{y}), \tag{6.35}$$

$$|f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})|^2 = f_{2\operatorname{Re}(A),2\operatorname{Re}(B),2\operatorname{Re}(C),|D|^2}^{(1)}(\boldsymbol{\omega},\mathbf{x}), \tag{6.36}$$

$$|f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})|^2 = f_{2\operatorname{Re}(A),2\operatorname{Re}(B),2\operatorname{Re}(C),|D|^2}^{(2)}(\boldsymbol{\omega},\mathbf{y}). \tag{6.37}$$

By substituting $A,B,C,D \to 2A,2B,2C,D^2$ into (6.28) (it's possible since $\mathrm{Re}\,(1-4(2A)) > 0$), we compute the first expectation in (6.34) as:

$$\mathbb{E}\left(f^{(1)}_{2A,2B,2C,D^2}(\boldsymbol{\omega},\mathbf{x})f^{(2)}_{2A,2B,2C,D^2}(\boldsymbol{\omega},\mathbf{y})\right) = D^4\left(\sqrt{1-8A}\right)^{-d}$$
$$\times \exp\left(\frac{4B^2}{2(1-8A)}\|\mathbf{x}+t\mathbf{y}\|^2 + 2C(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2)\right).$$

Next, we express B,C,D through A,t using (6.12):

$$\mathbb{E}\left(f^{(1)}_{2A,2B,2C,D^2}(\boldsymbol{\omega},\mathbf{x})f^{(2)}_{2A,2B,2C,D^2}(\boldsymbol{\omega},\mathbf{y})\right) = \left(\sqrt{\frac{(1-4A)^2}{1-8A}}\right)^d$$
$$\times \exp\left(\frac{4t(1-4A)}{2(1-8A)}\|\mathbf{x}+t\mathbf{y}\|^2 - (t+1)(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2)\right)$$
$$= \alpha_1 \exp\left(\alpha_2\|\mathbf{x}+t\mathbf{y}\|^2 - (t+1)\left(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2\right)\right), \tag{6.38}$$

By substituting $A,B,C,D \to 2\,\mathrm{Re}\,(A), 2\,\mathrm{Re}\,(B), 2\,\mathrm{Re}\,(C), |D|^2$ into (6.28) (it's possible since $\mathrm{Re}\,(1-8A) > 0$ and, hence, $\mathrm{Re}\,(1-8\,\mathrm{Re}\,(A)) > 0$), we can compute the second expectation in (6.34):

$$\mathbb{E}\left(f^{(1)}_{2\mathrm{Re}(A),2\mathrm{Re}(B),2\mathrm{Re}(C),|D|^2}(\boldsymbol{\omega},\mathbf{x})f^{(2)}_{2\mathrm{Re}(A),2\mathrm{Re}(B),2\mathrm{Re}(C),|D|^2}(\boldsymbol{\omega},\mathbf{y})\right) = |D|^4\left(\sqrt{1-8\,\mathrm{Re}\,(A)}\right)^{-d}$$
$$\cdot \exp\left(\frac{(B+\overline{B})^2}{2(1-8\,\mathrm{Re}\,(A))}\|\mathbf{x}+t\mathbf{y}\|^2 + 2\,\mathrm{Re}\,(C)\,(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2)\right).$$

Next, we observe that $|D|^4 = D^2\overline{D^2}$, $(B+\overline{B})^2 = B^2+\overline{B^2}+2|B^2|$ and use (6.12) to express B,C,D through A and C:

$$\mathbb{E}\left(f^{(1)}_{2\mathrm{Re}(A),2\mathrm{Re}(B),2\mathrm{Re}(C),|D|^2}(\boldsymbol{\omega},\mathbf{x})f^{(2)}_{2\mathrm{Re}(A),2\mathrm{Re}(B),2\mathrm{Re}(C),|D|^2}(\boldsymbol{\omega},\mathbf{y})\right)$$
$$= \left(\frac{(1-4A)(\overline{1-4A})}{1-8\,\mathrm{Re}\,(A)}\right)^{d/2}$$
$$\times \exp\left(\frac{s(2-8\,\mathrm{Re}\,(A))+2|1-4A|}{2(1-8\,\mathrm{Re}\,(A))}\|\mathbf{x}+t\mathbf{y}\|^2 - (t+1)(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2)\right)$$
$$= \alpha_3 \exp\left(\alpha_4\|\mathbf{x}+t\mathbf{y}\|^2 - (t+1)\left(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2\right)\right) \tag{6.39}$$

where in the last transition we also take into account that $(1-4A)(\overline{1-4A}) = 1-8\,\mathrm{Re}\,(A)+16|A|^2$. (6.30, 6.31, 6.32, 6.33, 6.34, 6.35-6.37, 6.38, 6.39) taken together result in (6.13). $\square$

Proof of Theorem 15

Proof. When A is real and $t = +1$, variance (6.13) has a form:

$$\begin{aligned}\mathrm{Var}_{p_{\mathrm{SG}}}\left(f_{\mathrm{GE}}^{(1)}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{GE}}^{(2)}(\boldsymbol{\omega},\mathbf{y})\right) &= \left(\frac{1-4A}{\sqrt{1-8A}}\right)^d \\ &\cdot \exp\left(\frac{2(1-4A)}{1-8A}\|\mathbf{x}+\mathbf{y}\|^2 - 2(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2)\right) - K(\mathbf{x},\mathbf{y})^2 \\ &= 2^{-d}\left(\frac{\rho+1}{\sqrt{\rho}}\right)^d \exp\left((1+\rho)\|\mathbf{x}+\mathbf{y}\|^2 - 2(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2)\right) - K(\mathbf{x},\mathbf{y})^2\end{aligned}$$

where we change the variable $\rho = \frac{1}{1-8A} \in (0,+\infty)$. We see that the minimum of variance with respect to $\rho \in (0,+\infty)$ coincides with the minimum of the logarithm of the first term:

$$g(\rho) = -d\log 2 + d\log(\rho+1) - \frac{d}{2}\log\rho + (1+\rho)\|\mathbf{x}+\mathbf{y}\|^2 - 2(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2).$$

All stationary points ρ^* can be found by setting its derivative to zero:

$$g'(\rho^*) = \frac{d}{\rho^*+1} - \frac{d}{2\rho^*} + \|\mathbf{x}+\mathbf{y}\|^2 = 0.$$

Multiply by $2\rho^*(\rho^*+1) > 0$ and obtain an equivalent quadratic equation:

$$\begin{gathered} d(\rho^*-1) + 2\rho^*(\rho^*+1)\|\mathbf{x}+\mathbf{y}\|^2 = 0; \\ 2\|\mathbf{x}+\mathbf{y}\|^2(\rho^*)^2 + (2\|\mathbf{x}+\mathbf{y}\|^2+d)\rho^* - d = 0; \\ \rho^*_{1,2} = \frac{1}{4\|\mathbf{x}+\mathbf{y}\|^2}\left(\pm\sqrt{(2\|\mathbf{x}+\mathbf{y}\|^2+d)^2 + 8d\|\mathbf{x}+\mathbf{y}\|^2} - 2\|\mathbf{x}+\mathbf{y}\|^2 - d\right). \end{gathered} \tag{6.40}$$

The root ρ_2^* of the quadratic equation with "$-$" sign in place of "$\pm$" (6.40) is a negative number. Since $\|\mathbf{x}+\mathbf{y}\|^2 > 0$, we conclude that the only stationary point is the positive root $\rho^* = \rho_1^* > 0$ with "$+$" sign in place of "$\pm$".

$g'(\rho)$ is a continuous function with $g'(\rho) \to -\infty$ as $\rho \to +0$ and $g'(\rho) \to \|\mathbf{x}+\mathbf{y}\|^2 > 0$ as $\rho \to +\infty$. There is only one ρ^* such that $g'(\rho^*) = 0$, and therefore for all $\rho < \rho^*$, $g'(\rho) < 0$ and for all $\rho > \rho^*$, $g'(\rho) < 0$. Hence, ρ^* is a global minimum of $g(\rho)$.

Since $g'(1) = \|\mathbf{x}+\mathbf{y}\|^2 > 0$, we also point out that $\rho^* < 1$. □

Proof of Theorem 16

Proof. First, we use $\mathrm{Var} Z = \mathbb{E}Z^2 - (\mathbb{E}Z)^2$ which holds for any random variable Z, e.g. $\mathrm{Re}\left(f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{y})\right)$:

$$\mathrm{Var}_{p_{\mathrm{pois}}(\boldsymbol{\omega})}\left(f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{y})\right) = \mathbb{E}\left(f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{x})^2 f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{y})^2\right) - \left(\mathbb{E}\left(f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{y})\right)\right)^2. \tag{6.41}$$

We know that $\left(\mathbb{E}\left(f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{y})\right)\right)^2 = K(\mathbf{x},\mathbf{y})^2$. Since $\boldsymbol{\omega}_1,\ldots,\boldsymbol{\omega}_d$ are independent, $f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{x})^2 f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{y})^2$ can be decomposed into a product of d independent random variables:

$$f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{x})^2 f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{y})^2 = \exp(2\lambda d - \|\mathbf{x}\|^2 - \|\mathbf{y}\|^2)\prod_{l=1}^{d}(\mathbf{x}_l\mathbf{y}_l)^{2\boldsymbol{\omega}_l}\lambda^{-2\boldsymbol{\omega}_l}.$$

Its expectation is therefore a product of d independent expectations:

$$\mathbb{E}\left(f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{x})^2 f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{y})^2\right) = \exp(2\lambda d - \|\mathbf{x}\|^2 - \|\mathbf{y}\|^2)\prod_{l=1}^{d}\mathbb{E}(\mathbf{x}_l\mathbf{y}_l)^{2\boldsymbol{\omega}_l}\lambda^{-2\boldsymbol{\omega}_l}.$$

We compute each expectation in the product. First, we rewrite it as a sum:

$$\mathbb{E}(\mathbf{x}_l\mathbf{y}_l)^{2\boldsymbol{\omega}_l}\lambda^{-2\boldsymbol{\omega}_l} = \sum_{k=0}^{\infty} p_k(\mathbf{x}_l\mathbf{y}_l)^{2k}\lambda^{-2k} = e^{-\lambda}\sum_{k=0}^{\infty}\frac{\lambda^k}{k!}(\mathbf{x}_i\mathbf{y}_i)^{2k}\lambda^{-2k} = e^{-\lambda}\sum_{k=0}^{\infty}\frac{(\mathbf{x}_l^2\mathbf{y}_l^2\lambda^{-1})^k}{k!}.$$

The last sum is a Taylor expansion of $\exp\left(\frac{\mathbf{x}_l^2\mathbf{y}_l^2}{\lambda}\right)$. So we have:

$$\mathbb{E}\left(f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{x})^2 f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{y})^2\right) = \exp\left(\lambda d + \lambda^{-1}\sum_{l=1}^{d}\mathbf{x}_l^2\mathbf{y}_l^2 - \|\mathbf{x}\|^2 - \|\mathbf{y}\|^2\right).$$

Taking it together with (6.41) results in (6.17). □

Proof of Theorem 17

Proof. The proof is similar to Theorem 16. First, we use $\mathrm{Var} Z = \mathbb{E}Z^2 - (\mathbb{E}Z)^2$ which holds for any random variable Z, e.g. $\mathrm{Re}\left(f_{\mathrm{geom}}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{geom}}(\boldsymbol{\omega},\mathbf{y})\right)$:

$$\mathrm{Var}_{p_{\mathrm{geom}}(\boldsymbol{\omega})}\left(f_{\mathrm{geom}}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{pois}}(\boldsymbol{\omega},\mathbf{y})\right) = \mathbb{E}\left(f_{\mathrm{geom}}(\boldsymbol{\omega},\mathbf{x})^2 f_{\mathrm{geom}}(\boldsymbol{\omega},\mathbf{y})^2\right) - \left(\mathbb{E}\left(f_{\mathrm{geom}}(\boldsymbol{\omega},\mathbf{x})f_{\mathrm{geom}}(\boldsymbol{\omega},\mathbf{y})\right)\right)^2. \tag{6.42}$$

We know that $\left(\mathbb{E}\left(f_{\text{pois}}(\boldsymbol{\omega},\mathbf{x})f_{\text{pois}}(\boldsymbol{\omega},\mathbf{y})\right)\right)^2 = K(\mathbf{x},\mathbf{y})^2$. Since $\boldsymbol{\omega}_1,\dots,\boldsymbol{\omega}_d$ are independent, $f_{\text{pois}}(\boldsymbol{\omega},\mathbf{x})^2 f_{\text{pois}}(\boldsymbol{\omega},\mathbf{y})^2$ can be decomposed into a product of d independent random variables:

$$f_{\text{geom}}(\boldsymbol{\omega},\mathbf{x})^2 f_{\text{geom}}(\boldsymbol{\omega},\mathbf{y})^2 = q^{-2d}\exp(-\|\mathbf{x}\|^2-\|\mathbf{y}\|^2)\prod_{l=1}^{d}(\boldsymbol{\omega}_l!)^{-2}((1-q)^{-1}\mathbf{x}_l\mathbf{y}_l)^{2\boldsymbol{\omega}_l}.$$

Its expectation is therefore a product of d independent expectations:

$$\mathbb{E}\left(f_{\text{geom}}(\boldsymbol{\omega},\mathbf{x})^2 f_{\text{geom}}(\boldsymbol{\omega},\mathbf{y})^2\right) = q^{-2d}\exp(-\|\mathbf{x}\|^2-\|\mathbf{y}\|^2)\prod_{l=1}^{d}\mathbb{E}(\boldsymbol{\omega}_l!)^{-2}((1-q)^{-1}\mathbf{x}_l\mathbf{y}_l)^{2\boldsymbol{\omega}_l}.$$

We compute each expectation in the product. First, we rewrite it as a sum:

$$\begin{aligned}\mathbb{E}(\boldsymbol{\omega}_l!)^{-2}((1-q)^{-1}\mathbf{x}_l\mathbf{y}_l)^{2\boldsymbol{\omega}_l} &= \sum_{k=0}^{\infty} p_k(k!)^{-2}((1-q)^{-1}\mathbf{x}_l\mathbf{y}_l)^{2k}\\ &= q\sum_{k=0}^{\infty}(k!)^{-2}((1-q)^{-1/2}\mathbf{x}_l\mathbf{y}_l)^{2k}.\end{aligned}$$

The last sum is a Taylor expansion of $I_0(2(1-q)^{-1/2}\mathbf{x}_l\mathbf{y}_l) = I_0(2(1-q)^{-1/2}|\mathbf{x}_l\mathbf{y}_l|)$ (I_0 is an even function). So we have:

$$\mathbb{E}\left(f_{\text{geom}}(\boldsymbol{\omega},\mathbf{x})^2 f_{\text{geom}}(\boldsymbol{\omega},\mathbf{y})^2\right) = q^{-d}\exp(-\|\mathbf{x}\|^2-\|\mathbf{y}\|^2)\prod_{l=1}^{d} I_0(2(1-q)^{-\frac{1}{2}}|\mathbf{x}_l\mathbf{y}_l|).$$

Taking it together with (6.42) results in (6.19). □

We take absolute values $|\mathbf{x}_l\mathbf{y}_l|$ instead of just $\mathbf{x}_l\mathbf{y}_l$ because the average of $\mathbf{x}_l^{(i)}$ and $\mathbf{y}_l^{(j)}$ would converge to zero due to different signs and wouldn't produce any meaningful statistic.

Proof of Theorem 19

Proof. We start with the case $M < d$. We factorize the variance of $\widehat{\mathscr{F}}_M^{\text{iid}}$ and $\widehat{\mathscr{F}}_M^{\text{ort}}$ by conditioning on the lengths of the used random samples. We have:

$$\begin{aligned}\widehat{\mathscr{F}}_M^{\text{iid}} = \int_{\mathbb{R}^d} \operatorname{Var}\left(\widehat{\mathscr{F}}_M^{\text{iid}} \,|\, \|\boldsymbol{\omega}^{(\text{iid},1)}\| = y_1,\dots,\|\boldsymbol{\omega}^{(\text{iid},M)}\| = y_M\right)\\ \times\prod_{m=1}^{M}\mathscr{P}(y_m)\mathrm{d}y_1\dots\mathrm{d}y_M.\end{aligned}$$

And, similarly:

$$\widehat{\mathscr{F}}_M^{\mathrm{ort}} = \int_{\mathbb{R}^d} \mathrm{Var}\left(\widehat{\mathscr{F}}_M^{\mathrm{ort}} \,|\, \|\boldsymbol{\omega}^{(\mathrm{ort},1)}\| = y_1, \ldots, \|\boldsymbol{\omega}^{(\mathrm{ort},M)}\| = y_M\right) \times \prod_{m=1}^{M} \mathscr{P}(y_m)\mathrm{d}y_1 \ldots \mathrm{d}y_M$$

where $\mathscr{P}$ is the probability density function for the distribution $\widetilde{\Omega}$. We use the fact that in both scenarios of i.i.d. samples and an block-orthogonal samples, the lengths of vectors $\boldsymbol{\omega}^{(\cdot,\cdot)}$ are sampled from the same distribution $\widetilde{\Omega}$ independently from their directions and from each other. Therefore, we have:

$$\mathrm{Var}(\widehat{\mathscr{F}}_M^{\mathrm{iid}}) - \mathrm{Var}(\widehat{\mathscr{F}}_M^{\mathrm{ort}}) = \int_{\mathbb{R}^d} T(y_1, \ldots, y_M) \prod_{m=1}^{M} \mathscr{P}(y_m)\mathrm{d}y_1 \ldots \mathrm{d}y_M, \tag{6.43}$$

where

$$T(y_1, \ldots, y_M) = \mathrm{Var}\left(\widehat{\mathscr{F}}_M^{\mathrm{iid}} \,|\, \|\boldsymbol{\omega}^{(\mathrm{iid},1)}\| = y_1, \ldots, \|\boldsymbol{\omega}^{(\mathrm{iid},M)}\| = y_M\right) - \mathrm{Var}\left(\widehat{\mathscr{F}}_M^{\mathrm{ort}} \,|\, \|\boldsymbol{\omega}^{(\mathrm{ort},1)}\| = y_1, \ldots, \|\boldsymbol{\omega}^{(\mathrm{ort},M)}\| = y_M\right)$$

Since the lengths of the samples are chosen independently from their directions, we conclude that:

$$\mathrm{Var}\left(\widehat{\mathscr{F}}_M^{\mathrm{iid}} \,|\, \|\boldsymbol{\omega}^{(\mathrm{iid},1)}\| = y_1, \ldots, \|\boldsymbol{\omega}^{(\mathrm{iid},M)}\| = y_M\right) = \mathrm{Var}\left(\frac{1}{M}\sum_{m=1}^{M} Z_m^{\mathrm{iid}}\right)$$

and

$$\mathrm{Var}\left(\widehat{\mathscr{F}}_M^{\mathrm{ort}} \,|\, \|\boldsymbol{\omega}^{(\mathrm{ort},1)}\| = y_1, \ldots, \|\boldsymbol{\omega}_M^{\mathrm{ort}}\| = y_M\}\right) = \mathrm{Var}\left(\frac{1}{M}\sum_{m=1}^{M} Z_m^{\mathrm{ort}}\right),$$

where $Z_m^{\mathrm{iid}} = \mathscr{G}_{y_m}((\boldsymbol{\omega}^{(\mathrm{iid},m)})^\top \mathbf{z}/\|\boldsymbol{\omega}^{(\mathrm{iid},m)}\|)$ and $Z_m^{\mathrm{ort}} = \mathscr{G}_{y_m}((\boldsymbol{\omega}^{(\mathrm{ort},m)})^\top \mathbf{z}/\|\boldsymbol{\omega}^{(\mathrm{ort},m)}\|)$.

Thus we have:

$$T(y_1, \ldots, y_M) = \mathrm{Var}\left(\frac{1}{M}\sum_{m=1}^{M} Z_m^{\mathrm{iid}}\right) - \mathrm{Var}\left(\frac{1}{M}\sum_{m=1}^{M} Z_m^{\mathrm{ort}}\right). \tag{6.44}$$

Now, by the similar analysis as in the proof of Theorem 11, we obtain for $\mathbf{g} \sim \mathscr{N}(0,1)^d$:

$$T(y_1, \ldots, y_M) \geq \frac{4}{M^2(d+2)} \sum_{m_1 < m_2} \sum_{k',k''=1}^{\infty} a_{2k'}(y_{m_1}) a_{2k''}(y_{m_2}) \|\mathbf{z}\|^{2k'+2k''} \mathbb{E}(\|\boldsymbol{\omega}\|^{2k'}) \mathbb{E}(\|\boldsymbol{\omega}\|^{2k''})$$

$$\times \frac{\mathbb{E}(\mathbf{g}_1^{2k'})\mathbb{E}(\mathbf{g}_1^{2k''})}{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{2k'}\right)\mathbb{E}\left(\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{2k''}\right)} = \frac{4}{M^2(d+2)}\sum_{m_1<m_2}$$

$$\left(\sum_{k=1}^{\infty} a_{2k}(y_{m_1})\|\mathbf{z}\|^{2k}\frac{\mathbb{E}(\|\boldsymbol{\omega}\|^{2k})\mathbb{E}(\mathbf{g}_1^{2k})}{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{2k}\right)}\right)\left(\sum_{k=1}^{\infty} a_{2k}(y_{m_2})\|\mathbf{z}\|^{2k}\frac{\mathbb{E}(\|\boldsymbol{\omega}\|^{2k})\mathbb{E}(\mathbf{g}_1^{2k})}{\mathbb{E}\left(\sqrt{\mathbf{g}_1^2+\cdots+\mathbf{g}_d^2}^{2t}\right)}\right)$$

$$= \frac{4}{M^2(d+2)}\sum_{m_1<m_2}(\mathscr{F}_{y_{m_1}}(\mathbf{z})-\mathscr{F}_{y_{m_1}}(\mathbf{0}_d))(\mathscr{F}_{y_{m_2}}(\mathbf{z})-\mathscr{F}_{y_{m_2}}(\mathbf{0}_d))$$

where $\mathscr{F}_y(\mathbf{z}) = \mathbb{E}\mathscr{G}(\mathbf{u}^\top\mathbf{z}, y)$, $\mathbf{u}\sim \mathrm{Unif}(\mathscr{S}^{d-1})$.

We conclude that:

$$\mathrm{Var}(\widehat{\mathscr{F}}_M^{\mathrm{iid}}) - \mathrm{Var}(\widehat{\mathscr{F}}_M^{\mathrm{ord}}) \geq \frac{4}{M^2(d+2)}\int_{\mathbb{R}^d}\sum_{m_1<m_2}(\mathscr{F}_{y_{m_1}}(\mathbf{z})-\mathscr{F}_{y_{m_1}}(\mathbf{0}_d))$$

$$\times(\mathscr{F}_{y_{m_2}}(\mathbf{z})-\mathscr{F}_{y_{m_2}}(\mathbf{0}_d))\prod_{m=1}^{M}\mathscr{P}(y_m)\mathrm{d}y_1\ldots\mathrm{d}y_M = \frac{4}{M^2(d+2)}\binom{M}{2}$$

$$\times\int_{\mathbb{R}^2}(\mathscr{F}_{y_1}(\mathbf{z})-\mathscr{F}_{y_1}(\mathbf{0}_d))(\mathscr{F}_{y_2}(\mathbf{z})-\mathscr{F}_{y_2}(\mathbf{0}_d))\mathscr{P}(y_1)\mathscr{P}(y_2)\mathrm{d}y_1\mathrm{d}y_2$$

$$= \frac{2(M-1)}{M(d+2)}(\mathscr{F}(\mathbf{z})-\mathscr{F}(\mathbf{0}_d))^2,$$

That completes the proof for the case $M\leq d$. The case $M>d$, M/d integer, is considered in the same way as in Theorem 11. □

Proof of Theorem 20

Proof. We have:

$$0\leq Z = \exp\left(-\left\|\sqrt{-A}\boldsymbol{\omega}-\frac{B}{2\sqrt{-A}}\mathbf{x}\right\|^2-\frac{B^2}{4A}\|\mathbf{x}\|^2+C\|\mathbf{x}\|^2\right)$$

$$\times\exp\left(-\left\|\sqrt{-A}\boldsymbol{\omega}-\frac{B}{2\sqrt{-A}}\mathbf{y}\right\|^2-\frac{B^2}{4A}\|\mathbf{y}\|^2+C\|\mathbf{y}\|^2\right)\leq\exp\left(-\frac{\|\mathbf{x}\|^2+\|\mathbf{y}\|^2}{4A}\right) \quad (6.45)$$

where the last inequality follows from $B=\sqrt{1-4A}$, $C=-1$ (Theorem 13).

Define: $Y = Z - \mathbb{E}[Z]$. Note that: $\mathbb{E}[Y] = 0$. Furthermore, from (6.45), we get: $-K(\mathbf{x},\mathbf{y}) \leq Y \leq \exp(-\frac{\|\mathbf{x}\|^2+\|\mathbf{y}\|^2}{4A}) - K(\mathbf{x},\mathbf{y})$. The following is true:

$$\mathbb{P}(|\widehat{\mathbf{x}}_{\mathrm{GE}}^{\top}\widehat{\mathbf{y}}_{\mathrm{GE}} - K(\mathbf{x},\mathbf{y})| \geq \varepsilon) = \mathbb{P}\left(\frac{Y_1+\cdots+Y_M}{M} \geq \varepsilon\right) = \mathbb{P}(Y_1+\cdots+Y_M \geq M\varepsilon),$$

where $Y_1,\ldots,Y_M$ are independent copies of Y. We complete the proof of the first part of the theorem by applying Hoeffding's inequality (Hoeffding, 1994) for the equation above which can be applied since Y_m's are zero-mean and bounded.

The second part of the theorem follows directly from the exact same method as applied in the proof of Theorem 18, e.g. we condition on the lengths of the sampled vectors $\boldsymbol{\omega}^{(m)}$, combined again with the analysis from Theorem 10 but this time for higher moments. In this modification, we also use the simple bound $\Delta \geq 0$ instead of (4.28) from the proof of Theorem 10. A critical difference from PosRFs in this case is that GERFs with $A < 0, t = +1$ are bounded resulting in the moment generating function $\mathscr{M}_Z(\cdot)$ being well-defined. □

Proof of Theorem 21

Proof. The proof is similar to the proof of Claim 1 from (Rahimi and Recht, 2007). Note that in the standard self-attention mechanism, queries and keys are renormalized by the multiplicative factor: $d^{-1/4}$. Thus denote: $\mathbf{x} = d^{-1/4}\mathbf{Q}_i$ and $\mathbf{y} = d^{-1/4}\mathbf{K}_j$. Note that $\|\mathbf{x}\|, \|\mathbf{y}\| \leq d^{-1/4}R$. Consider a vector $\mathbf{z} = [\mathbf{x}^{\top}, \mathbf{y}^{\top}]^{\top} \in \mathbb{R}^{2d}$. Note that: $\|\mathbf{z}\| \leq \sqrt{2}d^{-1/4}R$. By the analogous analysis as in Claim 1, we cover the ball with the center at $\mathbf{0}_{2d}$ and radius $\sqrt{2}d^{-1/4}R$ with ε-net of at most $\left(\frac{4\rho}{c}\right)^{2d}$ balls of radius c for $\rho = \sqrt{2}d^{-1/4}R$. If L_f denotes the Lipschitz constant of f (in the notation from (Rahimi and Recht, 2007)), the straightforward calculations lead to:

$$\mathbb{E}L_f^2 \leq \max_{\mathbf{x},\mathbf{y}} \exp\left(-\frac{1}{2}\left(1+\frac{1}{2A}\right)(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2)\right) \max_{\mathbf{x},\mathbf{y}} \left(2\|\mathbf{x}\|^2 + 2\|\mathbf{y}\|^2 + 4\mathbb{E}\|\boldsymbol{\omega}\|^2\right),$$

where $\boldsymbol{\omega} \sim \mathscr{N}(0,1)^d$. Thus we have: $\mathbb{E}L_f^2 \leq \gamma^2$, where: $\gamma = \sqrt{b_1\left(\frac{R^2}{\sqrt{d}} + d^2\right)}$. Using Theorem 20, we also notice that we can get analogous inequality as (6) from the proof of Claim 1 in (Rahimi and Recht, 2007), but for (reusing the notation from (Rahimi and Recht, 2007)) $D = 4M\max_{\mathbf{x},\mathbf{y}} \exp\left(\frac{3(\|\mathbf{x}\|^2+\|\mathbf{y}\|^2)}{2A}\right) = 4M\exp\left(\frac{3R^2}{A\sqrt{d}}\right)$. Thus, replacing (a) σ_p with γ, (b) D with $4M\exp\left(\frac{3R^2}{A\sqrt{d}}\right)$, (c) d with $2d$ and (d) $\mathrm{diam}(\mathscr{M})$ with b_2 in the statement of Claim 1, we obtain Theorem 21. □

Appendix 6.B Experimental details for Performer setups

Natural language processing

Pretraining was done on two publicly available datasets (Table 6.6). Following the original masked language modelling Transformer training (Devlin et al., 2018), we mask out 15% of random tokens in these two datasets, and train to predict the masked tokens as described in Section 2.2.3. We used the same hyperparameter setup for all baselines. The hyperparameters for pretraining are shown in Table 6.5.

Table 6.5 Hyperparameters for the models used in the natural language modelling experiment.

Parameter	Value
Number of heads (h)	12
Number of hidden layers (s)	12
Hidden layer size (d_{hid})	768
# of tokens (L)	$512+1$ (class token)
Batch size	256
M	256
Pretrain steps	1M
Dropout probability (Srivastava et al., 2014)	0.1
Optimizer	Adam (Kingma and Ba, 2015)
Learning rate	10^{-4}
Compute resources	64 TPUs

Table 6.6 Datasets used for pretraining in the natural language modelling experiment.

Dataset	# tokens	Average document length
Books (Zhu et al., 2015)	1.0B	37K
Wikipedia	3.1B	592

Speech modelling

Hyperparameters are reported in Table 6.7.

Image recognition

The image recognition experiments follow Section 4 in (He et al., 2021), where we use a ViT-Large (Table 6.11) and the same setup for training from scratch (Table 6.10) and uptraining (Table 6.9) as for the baseline from (He et al., 2021) trained with the exact self-attention

Table 6.7 Hyperparameters used in the speech modelling experiment.

Parameter	Value
Number of heads (h)	4
Number of hidden layers (s)	17
Hidden layer size (d_{hid})	256
# of tokens (L)	512
Batch size	256
Dropout probability (Srivastava et al., 2014)	0.1
Optimizer	Adam (Kingma and Ba, 2015)
Learning rate	10^{-4}
Compute resources	64 TPUs

(Table 6.8). Note that the fine-tuning setup has a shorter schedule which tests the adaptability of low-rank attention variants to the exact self-attention.

The ablations over sequence lengths are conducted by training from scratch and use ViT-tiny model (Table 6.12). Different sequence lengths are derived by adjusting the input size and the patch size which results in a different number of patches (Table 6.13).

Table 6.8 Hyperparameters used for pretraining in the image recognition experiment.

Parameter	Value
Batch size	4096
Optimizer	AdamW (Loshchilov and Hutter, 2017)
Base learning rate	1.5×10^{-4}
Weight decay (Loshchilov and Hutter, 2017)	0.05
Optimizer momentum (Loshchilov and Hutter, 2017)	$\beta_1, \beta_2 = 0.9, 0.95$
Learning rate schedule (Vaswani et al., 2017)	cosine decay
Warm up epochs (Vaswani et al., 2017)	40
Compute resources	64 TPUs

Table 6.9 Hyperparameters used for uptraining in the image recognition experiment.

Parameter	Value
Batch size	1024
Optimizer	AdamW (Loshchilov and Hutter, 2017)
Base learning rate	10^{-3}
Layer-wise learning rate decay	0.75
Weight decay (Loshchilov and Hutter, 2017)	0.05
Optimizer momentum (Loshchilov and Hutter, 2017)	$\beta_1, \beta_2 = 0.9, 0.999$
Learning rate schedule (Vaswani et al., 2017)	cosine decay
Warm up epochs (Vaswani et al., 2017)	5
Training epochs	50
Compute resources	64 TPUs

Table 6.10 Hyperparameters used for training from scratch in the image recognition experiment.

Parameter	Value
Batch size	4096
Optimizer	AdamW (Loshchilov and Hutter, 2017)
Base learning rate	10^{-4}
Layer-wise learning rate decay	0.75
Weight decay (Loshchilov and Hutter, 2017)	0.3
Optimizer momentum (Loshchilov and Hutter, 2017)	$\beta_1, \beta_2 = 0.9, 0.999$
Learning rate schedule (Vaswani et al., 2017)	cosine decay
Warm up epochs (Vaswani et al., 2017)	20
Training epochs	200
Compute resources	64 TPUs

Table 6.11 Parameters of ViT-Large.

Parameter	Value
Number of heads (h)	16
Number of layers (s)	24
Hidden layer size (d_{hid})	1024

Table 6.12 Parameters of ViT-Tiny.

Parameter	Value
Number of heads (h)	3
Number of layers (s)	12
Hidden layer size (d_{hid})	192

Table 6.13 ViT sequence length (number of patches) and the image input mapping.

Patches	Image input length L
8×8	$224 + 1$ (class token)
16×16	$224 + 1$ (class token)
32×32	$224 + 1$ (class token)
40×40	$240 + 1$ (class token)
44×44	$220 + 1$ (class token)

Chapter 7

Conclusions

7.1 Summary of contributions

This thesis is dedicated to a relatively young field (which emerged in 2020) of random feature self-attention approximation in long sequence Transformer networks. To our knowledge, this thesis summarizes all advances in this subject at the moment of writing. The main idea is to express the unnormalized self-attention matrix through a Gaussian kernel and use random features to approximate that kernel matrix as a randomized low-rank matrix which can be applied efficiently as a linear map (see Chapter 3). Notably, the approximation of the unnormalized self-attention is unbiased which allows the improvement of precision by increasing the number of samples, i.e. the number of random features. We start off using the well-known trigonometric random features (TrigRFs) with variance reduction through block-orthogonal random vectors in Chapter 3. We refer to the resulting block-orthogonal TrigRF-based mechanism for self-attention approximation as FAVOR (*F*ast *A*ttention *V*ia *O*rthogonal *R*andom features) and to the resulting efficient modification of Transformer as Performer.

Next, we encounter a problem that trigonometric random features lead to an approximation with negative matrix values. This leads to inconsistencies with the definition of self-attention as positive normalized weights and results in unstable training in some large-scale setups. We address the problem with a theoretical insight: positive random features (PosRFs) which allow an unbiased approximation of the Gaussian kernel which is strictly positive (Chapter 4). We provide a theoretical evaluation of these new random features, showing that block-orthogonal random vectors also reduce the variance in this case even in the non-asymptotic sense as it is for TrigRFs. However, we don't find a way to prove concentration bounds for PosRFs since these random features are unbounded. We refer to the block-orthogonal PosRF-based mechanism for self-attention approximation as FAVOR+

(positive FAVOR). We demonstrate that this mechanism can be trained in the language modelling setup where FAVOR is completely unstable and doesn't train at all.

Our final iteration over random feature improvements is chef's random tables and FAVOR++ in Chapter 6. We come up with many new types of random features for the Gaussian kernel which are all grouped under the name of chef's random tables (CRTs). CRTs consist of two subclasses: generalized exponential random features (GERFs) and discretely-induced random features (DIRFs). GERFs are a superclass containing TrigRFs and PosRFs as special cases. This class is parametrized by two scalar parameters. A subset of these parameters corresponds to a general family of positive-valued random features. By minimizing the variance of these positive-valued variants in a closed form, we discover optimal positive random features (OPRFs) which have a strictly smaller variance for Gaussian kernel estimation than PosRFs. DIRFs are based on the unbiased approximation of the Taylor series and intriguingly are induced by discrete random distributions rather than multivariate Gaussians as in GERFs. Special instantiations of the discrete probability distributions result in Poisson random features (PoisRFs) and geometric random features (GeomRFs). We evaluate all the newly proposed random feature variants and find that OPRFs work best among all variants including TrigRFs and PosRFs. Further, OPRFs appear to possess fruitful theoretical properties: positivity and boundedness. Positivity, in particular, allows proving strict variance improvements when using block-orthogonal random vectors similar to PosRFs. Using boundedness we can prove tight exponential concentration bounds around the Gaussian kernel and a uniform concentration around the unnormalized self-attention matrix when ORPFs are used in Transformers. Analogously to FAVOR and FAVOR+, we propose FAVOR++ which is a new mechanism for self-attention approximation based on OPRFs with block-orthogonal random vectors. We evaluate FAVOR++ and demonstrate its superior performance compared to its predecessor FAVOR+. We recommend using FAVOR++ for practitioners.

Finally, we show that the proposed low-rank self-attention mechanism has several intriguing extensions. First of all, the mechanism can be extended to the efficient approximation of causal self-attention when the unnormalized self-attention matrix is masked by a lower-triangular matrix of ones (Section 3.3). The generalization of that is taking arbitrary masks which can be applied efficiently as linear maps, resulting in masked self-attention (Section 4.5). Furthermore, we show that the random feature method can be extended to generalized attention (GA), where we use arbitrary mappings to produce low-rank matrices instead of those resulting in the unbiased approximation (Section 3.6). We also show that different types of random features can be combined into hybrid variants which can work efficiently in both small and large angle regimes (Section 6.2). Finally, we dedicate a lot of discussion to memory-efficient versions of Performers (SLiM Performers, Chapter 5) which have a very

small memory consumption under almost the same amount of computing. The idea is to use causal generalized self-attention and change the order of computations.

For each proposed method, this thesis presents extensive empirical evaluations in real-life large-scale learning setups and thorough theoretical analysis.

7.2 Open questions

We state the following open research questions about extensions of the method proposed in this thesis:

1. How can we further reduce the variance of Gaussian kernel estimation, both under the positivity restriction and not? Some concrete pointers in this direction are that

   (a) we still don't know the optimal closed form solution which minimizes the variance of GERFs in the most general case of complex A, B, C, D and $t \in \{-1, +1\}$ (6.13);

   (b) we don't know what is the optimal choice of the discrete distribution $\{p_k\}_{k=0}^{\infty}$ in DIRFs (Section 6.4) instead of special cases such as a Poisson or geometric distribution.

   These or other improvements can lead to a new iteration of efficient self-attention approximation mechanisms, e.g. FAVOR#.

2. For the existing random feature mechanisms such as TrigRFs, PosRFs, and OPRFs, can we improve concentration bounds discussed in this thesis, i.e. Theorems 3, 4, 6, 8, 19, 20, 21? In particular, can we improve upper bounds on the variance of block-orthogonal feature variants compared to i.i.d. variants (Theorems 3, 8, 19)? Can we get non-asymptotic variance improvements for TrigRFs (Theorem 3)? Can we improve concentration probability bounds by getting a multiplicative improvement in Theorem 8 instead of additive? Can we get left-tail concentration results for PosRFs and OPRFs? Currently, we can only provide right-tail concentration (Theorems 8, 19), that is guarantees for the approximated values not be too large. Perhaps, the novel concentration bound techniques developed in (Chamakh et al., 2020) could be useful for answering these questions.

3. The current version of CRTs only supports noncausal self-attention since the statistics (6.15,6.18,6.20) are aggregated over all values of $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(j)}$ and then used for random feature parameter inference. This is incompatible with the causal self-attention since, when processing $\mathbf{x}^{(i)} = d^{-1/4}\mathbf{Q}_i$, we cannot see the "future", namely elements

$\mathbf{y}^{(j)} = d^{-1/4}\mathbf{K}_j$ for $j > i$. Therefore, a natural open question is whether it is possible to extend CRTs to causal self-attention. One simple heuristic is to reuse statistics (6.15,6.18,6.20) from the previous instance or batch during training. However, is there a more holistic theory-guided approach?

4. Another fruitful direction is a search for new attention kernels under the umbrella of generalized attention (Section 3.6) which would result in a better downstream performance compared to the standard self-attention (Section 2.2.2). We have already given an example of the Performer-ReLU which outperforms the vanilla Transformer in the protein modelling task (Figures 3.9, 3.10), however these results don't generalize to other applications. Random features for some kernels were shown to be compatible for energy- and computation-efficient evaluation on optical processing units Wacker (2022). It would be, therefore, an interesting direction to evaluate such kernels in Transformers which are notoriously known for their high computation and energy costs Li et al. (2020).

We believe these theoretically-flavored open questions can have elegant solutions which would lead to improvements in impactful real-life applications of Performers.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lucic, M., and Schmid, C. (2021). ViViT: A video vision transformer. *CoRR*, abs/2103.15691.

Avron, H., Kapralov, M., Musco, C., Musco, C., Velingker, A., and Zandieh, A. (2017). Random Fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 253–262. PMLR.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bello, I., Zoph, B., Vaswani, A., Shlens, J., and Le, Q. V. (2019). Attention augmented convolutional networks. *CoRR*, abs/1904.09925.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Bitbol, A.-F., Dwyer, R. S., Colwell, L. J., and Wingreen, N. S. (2016). Inferring interaction partners from protein sequences. *Proceedings of the National Academy of Sciences*, 113(43):12180–12185.

Boffi, N. M., Tu, S., and Slotine, J. E. (2021). Nonparametric adaptive control and prediction: Theory and randomized algorithms. In *60th IEEE Conference on Decision and Control, CDC 2021, Austin, TX, USA, December 14-17, 2021*, pages 2935–2942. IEEE.

Bohanec, M. and Rajkovič, V. (1988). V.: Knowledge acquisition and explanation for multi-attribute decision. In *Making, 8 th International Workshop "Expert Systems and Their Applications*.

Bojarski, M., Choromanska, A., Choromanski, K., Fagan, F., Gouy-Pailler, C., Morvan, A., Sakr, N., Sarlós, T., and Atif, J. (2017). Structured adaptive and random spinners for fast machine learning computations. In Singh, A. and Zhu, X. J., editors, *Proceedings*

of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA, volume 54 of *Proceedings of Machine Learning Research*, pages 1020–1029. PMLR.

Bottou, L., Curtis, F. E., and Nocedal, J. (2016). Optimization methods for large-scale machine learning.

Brandes, N., Ofer, D., Peleg, Y., Rappoport, N., and Linial, M. (2022). ProteinBERT: a universal deep-learning model of protein sequence and function. *Bioinformatics*, 38(8):2102–2110.

Brent, R. P. (1971). An algorithm with guaranteed convergence for finding a zero of a function. *Comput. J.*, 14:422–425.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, page 213–229, Berlin, Heidelberg. Springer-Verlag.

Chamakh, L., Gobet, E., and Szabó, Z. (2020). Orlicz random fourier features. *Journal of Machine Learning Research*, 21(145):1–37.

Chan, W., Saharia, C., Hinton, G. E., Norouzi, M., and Jaitly, N. (2020). Imputer: Sequence modelling via imputation and dynamic programming. *CoRR*, abs/2002.08926.

Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. (2011). Differentially private empirical risk minimization. *J. Mach. Learn. Res.*, 12:1069–1109.

Chelba, C., Chen, M. X., Bapna, A., and Shazeer, N. (2020). Faster transformer decoding: N-gram masked self-attention. *CoRR*, abs/2001.04589.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. (2014). One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2635–2639.

Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y., Liu, Z., Ma, S., Xu, C., Xu, C., and Gao, W. (2021). Pre-trained image processing transformer. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12294–12305.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31, pages 6571–6583. Curran Associates, Inc.

Child, R., Gray, S., Radford, A., and Sutskever, I. (2019). Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Cho, Y. and Saul, L. K. (2009). Kernel methods for deep learning. In Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, pages 342–350. Curran Associates, Inc.

Choromanska, A., Choromanski, K., Bojarski, M., Jebara, T., Kumar, S., and LeCun, Y. (2016). Binary embeddings with structured hashed projections. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 344–353. JMLR.org.

Choromanski, K., Chen, H., Lin, H., Ma, Y., Sehanobish, A., Jain, D., Ryoo, M. S., Varley, J., Zeng, A., Likhosherstov, V., Kalashnikov, D., Sindhwani, V., and Weller, A. (2022a). Hybrid random features. In *International Conference on Learning Representations (ICLR)*.

Choromanski, K., Downey, C., and Boots, B. (2018a). Initialization matters: Orthogonal predictive state recurrent neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Choromanski, K., Lin, H., Chen, H., Zhang, T., Sehanobish, A., Likhosherstov, V., Parker-Holder, J., Sarlos, T., Weller, A., and Weingarten, T. (2022b). From block-toeplitz matrices to differential equations on graphs: towards a general theory for scalable masked transformers. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 3962–3983. PMLR.

Choromanski, K., Rowland, M., Sindhwani, V., Turner, R. E., and Weller, A. (2018b). Structured evolution with compact architectures for scalable policy optimization. In Dy, J. G. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 969–977. PMLR.

Choromanski, K. and Sindhwani, V. (2016). Recycling randomness with structure for sublinear time kernel expansions. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2502–2510. JMLR.org.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A.

(2021). Rethinking attention with performers. In *International Conference on Learning Representations*.

Choromanski, K. M., Rowland, M., and Weller, A. (2017a). The unreasonable effectiveness of structured random orthogonal embeddings. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 219–228.

Choromanski, K. M., Rowland, M., and Weller, A. (2017b). The unreasonable effectiveness of structured random orthogonal embeddings. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 219–228.

Cong, Q., Anishchenko, I., Ovchinnikov, S., and Baker, D. (2019). Protein interaction networks revealed by proteome coevolution. *Science*, 365(6449):185–189.

Consortium, U. (2019). Uniprot: a worldwide hub of protein knowledge. *Nucleic acids research*, 47(D1):D506–D515.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, 3rd Edition*. MIT Press.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 1 edition.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q., and Salakhutdinov, R. (2019). 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.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). 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. Association for Computational Linguistics.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Du, Y., Meier, J., Ma, J., Fergus, R., and Rives, A. (2020). Energy-based models for atomic-resolution protein conformations. *arXiv preprint arXiv:2004.13167*.

Dua, D. and Graff, C. (2017a). Banknote authentication data set, UCI machine learning repository.

Dua, D. and Graff, C. (2017b). Chess (king-rook vs. king) data set, UCI machine learning repository.

Dua, D. and Graff, C. (2017c). UCI machine learning repository.

Elnaggar, A., Heinzinger, M., Dallago, C., and Rost, B. (2019). End-to-end multitask learning, from protein language to protein features without alignments. *bioRxiv*, page 864405.

Fedus, W., Zoph, B., and Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Frostig, R., Johnson, M., and Leary, C. (2018). Compiling machine learning programs via high-level tracing. In *Conference on Machine Learning and Systems 2018*.

Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biol. Cybern.*, 20(3–4):121–136.

Girdhar, R., João Carreira, J., Doersch, C., and Zisserman, A. (2019). Video action transformer network. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 244–253.

Gong, H., Chen, G., Liu, S., Yu, Y., and Li, G. (2021a). Cross-modal self-attention with multi-task pre-training for medical visual question answering. In *Proceedings of the 2021 International Conference on Multimedia Retrieval*, ICMR '21, page 456–460, New York, NY, USA. Association for Computing Machinery.

Gong, Y., Chung, Y., and Glass, J. R. (2021b). AST: audio spectrogram transformer. *CoRR*, abs/2104.01778.

Gonon, L. (2021). Random feature neural networks learn Black-Scholes type PDEs without curse of dimensionality. *CoRR*, abs/2106.08900.

Griewank, A. (1992). Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optimization Methods and Software*, 1(1):35–54.

Griewank, A. and Walther, A. (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Second Edition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104).

Gulati, A., Qin, J., Chiu, C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., and Pang, R. (2020). Conformer: Convolution-augmented transformer for speech recognition. In Meng, H., Xu, B., and Zheng, T. F., editors, *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 5036–5040. ISCA.

Guo, M., Cai, J., Liu, Z., Mu, T., Martin, R. R., and Hu, S. (2020). PCT: point cloud transformer. *CoRR*, abs/2012.09688.

Han, I., Avron, H., Shoham, N., Kim, C., and Shin, J. (2021). Random features for the neural tangent kernel. *CoRR*, abs/2104.01351.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2021). Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Heinzerling, B. and Strube, M. (2019). Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 273–291, Florence, Italy. Association for Computational Linguistics.

Hendrycks, D. and Gimpel, K. (2016). Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hoeffding, W. (1994). *Probability Inequalities for sums of Bounded Random Variables*, pages 409–426. Springer New York, New York, NY.

Hopf, T. A., Colwell, L. J., Sheridan, R., Rost, B., Sander, C., and Marks, D. S. (2012). Three-dimensional structures of membrane proteins from genomic sequencing. *Cell*, 149(7):1607–1621.

Horton, P. and Nakai, K. (1996). A probabilistic classification system for predicting the cellular localization sites of proteins. In *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, page 109–115. AAAI Press.

Hsu, W., Bolte, B., Tsai, Y. H., Lakhotia, K., Salakhutdinov, R., and Mohamed, A. (2021). Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *CoRR*, abs/2106.07447.

Ingraham, J., Garg, V., Barzilay, R., and Jaakkola, T. (2019). Generative models for graph-based protein design. In *Advances in Neural Information Processing Systems*, pages 15794–15805.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020). Transformers are RNNs: Fast autoregressive transformers with linear attention. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kitaev, N., Kaiser, L., and Levskaya, A. (2020). Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Kovaleva, O., Romanov, A., Rogers, A., and Rumshisky, A. (2019). Revealing the dark secrets of BERT. *arXiv preprint arXiv:1908.08593*.

Krogh, A. and Hertz, J. A. (1991). A simple weight decay can improve generalization. In *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91, page 950–957, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ladner, R. E. and Fischer, M. J. (1980). Parallel prefix computation. *J. ACM*, 27(4):831–838.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020). ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Laparra, V., Gonzalez, D. M., Tuia, D., and Camps-Valls, G. (2015). Large-scale random features for kernel regression. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 17–20.

Lee, D. (1986). Fast multiplication of a recursive block toeplitz matrix by a vector and its application. *J. Complex.*, 2(4):295–305.

Li, B., Pandey, S., Fang, H., Lyv, Y., Li, J., Chen, J., Xie, M., Wan, L., Liu, H., and Ding, C. (2020). Ftrans: Energy-efficient acceleration of transformers using fpga. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '20, page 175–180, New York, NY, USA. Association for Computing Machinery.

Li, Z., Ton, J., Oglic, D., and Sejdinovic, D. (2021). Towards a unified analysis of random Fourier features. *J. Mach. Learn. Res.*, 22:108:1–108:51.

Likhosherstov, V., Choromanski, K., Dubey, A., Liu, F., Sarlos, T., and Weller, A. (2022). Chefs' random tables: Non-trigonometric random features.

Likhosherstov, V., Choromanski, K. M., Davis, J. Q., Song, X., and Weller, A. (2021). Sub-linear memory: How to make performers SLiM. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 6707–6719. Curran Associates, Inc.

Lim, T. S., Loh, W.-Y., and Shih, Y.-S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40:203–228.

Lin, H., Chen, H., Choromanski, K. M., Zhang, T., and Laroche, C. (2020). Demystifying orthogonal Monte Carlo and beyond. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Liu, Y. and Lapata, M. (2019). Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arxiv:1907.11692.

Loshchilov, I. and Hutter, F. (2017). Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Luo, H., Zhang, S., Lei, M., and Xie, L. (2020). Simplified self-attention for transformer-based end-to-end speech recognition. *CoRR*, abs/2005.10463.

Luo, S., Li, S., Cai, T., He, D., Peng, D., Zheng, S., Ke, G., Wang, L., and Liu, T.-Y. (2021). Stable, fast and accurate: Kernelized attention with relative positional encoding. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*.

Madani, A., McCann, B., Naik, N., Keskar, N. S., Anand, N., Eguchi, R. R., Huang, P., and Socher, R. (2020). Progen: Language modeling for protein generation. *CoRR*, abs/2004.03497.

Minh, H. Q. (2016). Operator-valued Bochner theorem, Fourier feature maps for operator-valued kernels, and vector-valued learning. *CoRR*, abs/1608.05639.

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142.

Nash, W. J. and Tasmania. (1994). *The Population biology of abalone (Haliotis species) in Tasmania. 1, Blacklip abalone (H. rubra) from the north coast and the islands of Bass Strait / Warwick J. Nash ... [et al.]*. Sea Fisheries Division, Dept. of Primary Industry and Fisheries, Tasmania Hobart.

Olave, M., Rajkovic, V., and Bohanec, M. (1989). An application for admission in public school systems. *Expert Systems in Public Administration*, 1:145–160.

Oliva, J. B., Neiswanger, W., Póczos, B., Xing, E. P., Trac, H., Ho, S., and Schneider, J. G. (2015). Fast function to function regression. In Lebanon, G. and Vishwanathan, S. V. N., editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, volume 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org.

Ott, M., Edunov, S., Grangier, D., and Auli, M. (2018). Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

Ovchinnikov, S., Kamisetty, H., and Baker, D. (2014). Robust and accurate prediction of residue–residue interactions across protein interfaces using evolutionary information. *Elife*, 3:e02030.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pages 5206–5210. IEEE.

Parisotto, E., Song, F., Rae, J., Pascanu, R., Gulcehre, C., Jayakumar, S., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., Botvinick, M., Heess, N., and Hadsell, R. (2020). Stabilizing transformers for reinforcement learning. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7487–7498. PMLR.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. (2018). Image transformer. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4052–4061. PMLR.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

Qi, D., Su, L., Song, J., Cui, E., Bharti, T., and Sacheti, A. (2020). Imagebert: Cross-modal pre-training with large-scale weak-supervised image-text data. *CoRR*, abs/2001.07966.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C., and Lillicrap, T. P. (2020). Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.

Rahimi, A. and Recht, B. (2008a). Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, Los Alamitos, CA, USA. IEEE Computer Society.

Rahimi, A. and Recht, B. (2008b). Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1313–1320. Curran Associates, Inc.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR.

Rives, A., Goyal, S., Meier, J., Guo, D., Ott, M., Zitnick, C., Ma, J., and Fergus, R. (2019). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioArxiv*.

Rohra, J., Perumal, B., J.N., S., Thakur, P., and Bhatt, R. (2017). *User Localization in an Indoor Environment Using Fuzzy Hybrid of Particle Swarm Optimization and Gravitational Search Algorithm with Neural Networks*, pages 286–295.

Rowland, M., Hron, J., Tang, Y., Choromanski, K., Sarlós, T., and Weller, A. (2019). Orthogonal estimation of wasserstein distances. In Chaudhuri, K. and Sugiyama, M., editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 186–195. PMLR.

Roy, A., Saffar, M., Vaswani, A., and Grangier, D. (2020). Efficient content-based sparse attention with routing transformers. *CoRR*, abs/2003.05997.

Rudin, W. (2017). *Fourier Analysis on Groups*. Dover Books on Mathematics. Dover Publications.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. (2019). Megatron-lm: Training multi-billion parameter language models using model parallelism. cite arxiv:1909.08053.

Sriperumbudur, B. K. and Szabó, Z. (2015). Optimal rates for random Fourier features. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1144–1152.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Sun, C., Myers, A., Vondrick, C., Murphy, K., and Schmid, C. (2019). VideoBERT: A joint model for video and language representation learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7463–7472.

Sun, Y., Gilbert, A. C., and Tewari, A. (2018). But how does it work in theory? Linear SVM with random features. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3383–3392.

Sutherland, D. J. and Schneider, J. G. (2015). On the error of random Fourier features. In Meila, M. and Heskes, T., editors, *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015, July 12-16, 2015, Amsterdam, The Netherlands*, pages 862–871. AUAI Press.

Tan, H. and Bansal, M. (2019). LXMERT: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, Hong Kong, China. Association for Computational Linguistics.

Tsai, Y.-H. H., Bai, S., Yamada, M., Morency, L.-P., and Salakhutdinov, R. (2019). Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4335–4344.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Vig, J. (2019). A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*.

Vig, J. and Belinkov, Y. (2019). Analyzing the structure of attention in a transformer language model. *CoRR*, abs/1906.04284.

Vig, J., Madani, A., Varshney, L. R., Xiong, C., Socher, R., and Rajani, N. F. (2020). Bertology meets biology: Interpreting attention in protein language models. *CoRR*, abs/2006.15222.

Wacker, J. (2022). *Random features for dot product kernels and beyond*. PhD thesis. EURECOM. Personal use of this material is permitted. The definitive version of this paper was published in Thesis and is available at :.

Walz, G. (2016). *Lexikon der Mathematik: Band 2: Eig bis Inn*. Springer-Verlag.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020). Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768.

Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, 26(4):359–372.

Weigt, M., White, R. A., Szurmant, H., Hoch, J. A., and Hwa, T. (2009). Identification of direct residue contacts in protein–protein interaction by message passing. *Proceedings of the National Academy of Sciences*, 106(1):67–72.

Wu*, Z., Liu*, Z., Lin, J., Lin, Y., and Han, S. (2020). Lite transformer with long-short range attention. In *International Conference on Learning Representations*.

Xiao, T., Li, Y., Zhu, J., Yu, Z., and Liu, T. (2019). Sharing attention weights for fast transformer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5292–5298. ijcai.org.

Xie, J., Liu, F., Wang, K., and Huang, X. (2019). Deep kernel learning via random Fourier features. *CoRR*, abs/1910.02660.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T.-Y. (2020). On layer normalization in the transformer architecture.

Yamada, I., Asai, A., Shindo, H., Takeda, H., and Matsumoto, Y. (2020). LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

Yang, F., Yang, H., Fu, J., Lu, H., and Guo, B. (2020). Learning texture transformer network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yang, J., Sindhwani, V., Fan, Q., Avron, H., and Mahoney, M. W. (2014). Random laplace feature maps for semigroup kernels on histograms. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 971–978. IEEE Computer Society.

Yang, T., Li, Y., Mahdavi, M., Jin, R., and Zhou, Z. (2012). Nyström method vs random Fourier features: A theoretical and empirical comparison. In Bartlett, P. L., Pereira, F. C. N., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 485–493.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc.

Yu, F. X., Suresh, A. T., Choromanski, K. M., Holtmann-Rice, D. N., and Kumar, S. (2016). Orthogonal random features. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1975–1983.

Zhang, J., He, T., Sra, S., and Jadbabaie, A. (2020a). Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*.

Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., Gao, J., Liu, J., and Dolan, B. (2020b). DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.

Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. (2021). Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268.

Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2021). Deformable DETR: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *IEEE international conference on computer vision*, pages 19–27.