RETHINKING ATTENTION WITH PERFORMERS

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

ABSTRACT

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

1 INTRODUCTION AND RELATED WORK

Transformers (Vaswani et al., 2017; Dehghani et al., 2019) are powerful neural network architectures that have become SOTA in several areas of machine learning including natural language processing (NLP) (e.g. speech recognition (Luo et al., 2020)), neural machine translation (NMT) (Chen et al., 2018), document generation/summarization, time series prediction, generative modeling (e.g. image generation (Parmar et al., 2018)), music generation (Huang et al., 2019), and bioinformatics (Rives et al., 2019; Madani et al., 2020; Ingraham et al., 2019; Elnaggar et al., 2019; Du et al., 2020).

Transformers rely on a trainable *attention* mechanism that identifies complex dependencies between the elements of each input sequence. Unfortunately, the regular Transformer scales quadratically with the number of tokens L in the input sequence, which is prohibitively expensive for large L and precludes its usage in settings with limited computational resources even for moderate values of L. Several solutions have been proposed to address this issue (Beltagy et al., 2020; Gulati et al., 2020; Chan et al., 2020; Child et al., 2019; Bello et al., 2019). 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). There is also a long line of research on using dense attention matrices, but defined by low-rank kernels substituting softmax (Katharopoulos et al., 2020; Shen et al., 2018). Those methods critically rely on kernels admitting explicit representations as dot-products of finite positive-feature vectors.

The approaches above 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 with sparse attention using more

[*]Equal contribution.

[1]Correspondence to {kchoro,lcolwell}@google.com.

Code for Transformer models on protein data can be found in github.com/google-research/google-research/tree/master/protein_lm and Performer code can be found in github.com/google-research/google-research/tree/master/performer.