FRONTAL LOW-RANK RANDOM TENSORS FOR HIGH-ORDER FEATURE REPRESENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Representing high-order (second-order or higher) information in deep neural networks is essential in many tasks such as fine-grained visual understanding and multi-modal information fusion. Bilinear models are often used to extract second-order information. As a basis, extracting higher-order information requires extra computation. In this paper, we propose an approach to representing high-order information via a simple yet effective bilinear form. Specifically, our contribution is two-fold: (1) From the multilinear perspective, we derive a bilinear form of low complexity, assuming that the three-way tensor has low-rank frontal slices. (2) Rather than learning the tensor entries from data, we sample the entries from different underlying distributions, and prove that the underlying distribution influences the information order. We perform temporal action segmentation experiments to evaluate our method. The results demonstrate that our bilinear form, employed as intermediate layers in deep neural networks, is computationally efficient; meanwhile it is effective as it achieves new state-of-the-art results on public benchmarks.

1 INTRODUCTION

A standard deep convolutional neural network mainly comprises convolutional layers and pooling layers. Despite effectiveness, such layers can only extract first-order information and hence are not suitable for fine-grained recognition and information fusion. It is reported in the literature that modeling second-order feature interactions (e.g. statistical covariance and feature co-occurrence) is essential for a wide range of visual tasks like fine-grained recognition (Lin et al., 2018b; Kong & Fowlkes, 2017), human action understanding (Girdhar & Ramanan, 2017; Cherian et al., 2017) and visual question answering (Kim et al., 2018; Fukui et al., 2016). When considering higher-order information, the extracted features are more discriminative and yield impressive performance (Cui et al., 2017; Koniusz et al., 2017).

Although modeling high-order feature interaction has many successful applications, most existing methods have a critical drawback: *The computational cost increases exponentially with respect to the information order and the input feature dimension.* Let us take bilinear pooling for second-order information extraction as an example. Given two feature vectors $\boldsymbol{x} \in \mathbb{R}^{D_X}$ and $\boldsymbol{y} \in \mathbb{R}^{D_Y}$, a generic bilinear model (Ben-Younes et al., 2017, Eq. (2)) is given by:

$$\boldsymbol{z} = \boldsymbol{\mathcal{T}} \times_1 \boldsymbol{x} \times_2 \boldsymbol{y}, \tag{1}$$

where $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{D_X \times D_Y \times D_Z}$ is a three-way tensor, and the operations $\times_1$ and $\times_2$ are the mode-1 and mode-2 multiplication, respectively. Despite being a powerful scheme, such a model tends to suffer from the *curse of dimensionality* and *intractable computation* (Bellman, 2015). Specifically, the number of free parameters in $\boldsymbol{\mathcal{T}}$ grows cubically as the feature dimensionality, i.e. $\mathcal{O}(D_X D_Y D_Z)$, hence limiting its use in large-scale scenarios.

To reduce the complexity, a common solution is to impose specific assumptions on the structure of $\boldsymbol{\mathcal{T}}$ in Eq. (1). In the work of Kim et al. (2016); Ben-Younes et al. (2017); Kong & Fowlkes (2017), low-rank assumption on $\boldsymbol{\mathcal{T}}$ is employed. In the work of Gao et al. (2016), count sketch is used to extract approximated second-order information, which can be represented by a shorter feature vector than the full second-order information. Although such conventional bilinear pooling methods can extract second-order information effectively, extracting higher-order information requires extra operations (Cui et al., 2017) and may further increase the computational cost.

In this paper, we aim at deriving a simple bilinear model, with which changing the order of information does not involve extra computation. Therefore, our investigations have two aspects: (1) From the multilinear perspective, we assume that each frontal slice of $\mathcal{T}$ — but not $\mathcal{T}$ itself — is a low-rank matrix. Then we derive a new bilinear form that first reduces the feature dimension and then performs vector outer product. Different from Kim et al. (2016) and Ben-Younes et al. (2017), that first lift the feature dimension and then perform Hadamard product, our method has runtime complexity $\mathcal{O}(D\sqrt{d}+d)$ and space complexity $\mathcal{O}(D\sqrt{d})$ compared to $\mathcal{O}(Dd+d)$ and $\mathcal{O}(Dd)$ in their works[1]. Thus, our bilinear operation is lightweight and can be employed in deep neural networks as intermediate layers in an efficient manner. (2) Rather than learning the model parameters from data, the entries of each frontal slice are determined by random projection matrices drawn from a specific distribution. Our key insight is that, while the low-rank structure allows a reduction in the number of parameters, the loss in representational power is compensated by the random projection matrices. In particular, we show that when these random matrices are sampled with different underlying distributions, the model approximates feature maps of different reproducing kernel Hilbert spaces (RKHSs). For example, when they are sampled from the Rademacher distribution, the model approximates the multiplication of linear kernels (cf. Theorem 1). When they are sampled from the Gaussian distribution with orthogonality constraints, the model approximates multiplication of Gaussian kernels (cf. Theorem 2). Hence, we can explicitly manipulate the model capacity without sacrificing the computational efficiency.

To verify the effectiveness of our method, we perform experiments on fine-grained action parsing, in which our bilinear model with random tensor entries is used as intermediate layers in temporal convolutional deep neural networks. The experimental results show that our method yields superior efficiency and performance to other related methods on different datasets.

Our contributions can be summarized as follows:

- We derive a novel bilinear model with a random three-way tensor. Using low-rank decomposition of each frontal slice, our method significantly reduces the number of parameters of the bilinear model, and hence can serve as a computationally efficient feature fusion operation.
- Based on different tensor entry distributions, we prove that the proposed random tensors can estimate the feature maps to reproducing kernel Hilbert spaces (RKHSs) with different compositional kernels. Therefore, the order of information is influenced by such underlying distributions.
- We combine our method with state-of-the-art deep neural networks for action segmentation and produce superior results.

2 Related Work

Representing high-order information in practice. Due to the computational cost, second-order information is mostly considered in various practical tasks, such as fine-grained image understanding (Carreira et al., 2012; Gao et al., 2016; Yu & Salzmann, 2018; Koniusz et al., 2017; Li et al., 2017a; Lin et al., 2015; 2018b;a; Kong & Fowlkes, 2017; Li et al., 2018; Wei et al., 2018; Gou et al., 2018), fine-grained action understanding (Feichtenhofer et al., 2016; Girdhar & Ramanan, 2017; Cherian et al., 2017; Zhang et al., 2019), visual question answering (Yu et al., 2018; Ben-Younes et al., 2017; Kim et al., 2018; Fukui et al., 2016) and beyond. In many studies, second-order information is extracted only once before the classification layer in a deep neural net (Yu & Salzmann, 2018; Lin et al., 2015; 2018b;a; Kong & Fowlkes, 2017; Wang et al., 2017; Diba et al., 2017; Feichtenhofer et al., 2016). For example, Lin et al. (2018b) compute the vector outer product to merge outputs from two individual CNN streams. Wei et al. (2018) transform CNN features to the compact Grassmann manifold using singular-value decomposition. Such methods either lift the feature dimension from D to D^2, or introduce expensive computation in both forward and backward passes, and hence are not applicable as intermediate layers in deep neural networks. Consequently, extracting higher-order ($\geq$ 3) information is even more challenging, although it is reported that higher-order information is more discriminative. Koniusz et al. (2017) compute higher-order feature occurrence in the *bag-of-words* pipeline, rather than in deep neural networks.

[1] D and d are input and output feature dimensions of the bilinear model, respectively; usually, $D \ll d$.

Low-rank tensor structure. There exist many investigations on low-rank tensor decomposition for bilinear models. For instance, Kim et al. (2016) assume that each frontal slice of the three-way tensor can be decomposed into two low-rank matrices, and the fusion of the two input features can then be achieved by matrix multiplication and Hadamard product. To improve the performance of visual question answering, Yu et al. (2017) introduce more operations after the low-rank bilinear pooling (Kim et al., 2016) such as dropout, power normalization, L2 normalization and so forth. Ben-Younes et al. (2017) rely on Tucker decomposition of the tensor, producing three matrices and a smaller core three-way tensor.

High-order information approximation. Representing high-order information explicitly can lead to combinatorial explosions. To avoid the *curse of dimensionality*, feature approximation methods based on *reproducing kernel Hilbert space* (RKHS) theories have been recently investigated. For example, Kar & Karnick (2012) use binary random entries to approximate inner product kernels, especially the p-th order polynomial kernels. Gao et al. (2016) and Pham & Pagh (2013) use tensor sketch to approximate polynomial kernels, which has lower approximation error bound but higher computational cost. Yu et al. (2016) use orthogonal random features to approximate feature maps of Gaussian kernels. To boost the computational speed, a structured version with normalized Walsh-Hadamard matrices is proposed. Such feature approximation methods also improve the efficiency of higher-order information extraction. For example, based on count sketch (Gao et al., 2016), Cui et al. (2017) extract 4th-order information from CNN features in an efficient manner.

Herein, we design a new bilinear form with lower complexity than Kim et al. (2016), Yu et al. (2017) and Ben-Younes et al. (2017). Also, it provides a generic framework to approximate feature maps of RKHSs with compositional kernels. Via sampling tensor entries from different distributions, our method shows that the output feature vector lies within certain RKHS, and hence we can manipulate the model capacity while retaining the same computational complexity.

3 Proposed Method

In this section we first introduce how to decrease the number of parameters in the bilinear model Eq. (1) via low-rank assumption. Afterwards, we present our investigations on how to approximate feature maps of kernels via random projection.

3.1 Tensor frontal low-rank approximation

Here we follow the tensor notations in Kolda & Bader (2009). Eq. (1) can be re-written in terms of matrix-vector multiplication as

$$\boldsymbol{z} = \boldsymbol{T}_{(3)}\, vec(\boldsymbol{x} \otimes \boldsymbol{y}), \tag{2}$$

where $\boldsymbol{T}_{(3)}$ is the mode-3 matricization of $\boldsymbol{\mathcal{T}}$, $vec(\cdot)$ denotes column-wise vectorization of a matrix, and $\otimes$ denotes vector outer product. In other words, the bilinear operation in Eq. (1) is equivalent to first computing the correlation matrix between the two features and then performing a linear projection.

It follows from Eq. (2) that each entry of the output feature vector $\boldsymbol{z}$ is a weighted sum of all the entries in the correlation matrix $\boldsymbol{x} \otimes \boldsymbol{y}$, i.e.,

$$z_k = \langle vec(\mathcal{T}[:,:,k]), vec(\boldsymbol{x} \otimes \boldsymbol{y})\rangle = \sum_{i=1}^{D_X}\sum_{j=1}^{D_Y} \mathcal{T}[i,j,k] v_{f(i,j)}, \tag{3}$$

where $\boldsymbol{v} := vec(\boldsymbol{x} \otimes \boldsymbol{y})$. If the frontal matrix $\mathcal{T}[:,:,k]$ is a rank-one matrix, i.e., $\mathcal{T}[:,:,k] = \boldsymbol{e} \otimes \boldsymbol{f}$ for some $\boldsymbol{e} \in \mathbb{R}^{D_X}$ and $\boldsymbol{f} \in \mathbb{R}^{D_Y}$, then we can rewrite Eq. (3) as $z_k = \langle vec(\boldsymbol{e} \otimes \boldsymbol{f}), vec(\boldsymbol{x} \otimes \boldsymbol{y})\rangle = \langle \boldsymbol{e}, \boldsymbol{x}\rangle\langle \boldsymbol{f}, \boldsymbol{y}\rangle$.

Thus, we define the projection matrices $\boldsymbol{E} = [\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_M]^T$ and $\boldsymbol{F} = [\boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_N]^T$ for two sets of vectors $\{\boldsymbol{e}_i\}_{i=1}^M \subset \mathcal{X}$ and $\{\boldsymbol{f}_j\}_{j=1}^N \subset \mathcal{Y}$ with $M \leq D_X$ and $N \leq D_Y$. Then, the fusion map $\phi : \mathbb{R}^{D_X} \times \mathbb{R}^{D_Y} \to \mathbb{R}^{MN}$ can be defined as

$$\boldsymbol{z} := \phi(\boldsymbol{x}, \boldsymbol{y}) = vec\left((\boldsymbol{E}\boldsymbol{x}) \otimes (\boldsymbol{F}\boldsymbol{y})\right). \tag{4}$$

If we assume further that $\mathcal{T}[:,:,k]$ is a rank-R matrix, i.e., $\mathcal{T}[:,:,k] = \sum_{r=1}^{R} \boldsymbol{e}_i^r \otimes \boldsymbol{f}_j^r$, we obtain

$$\boldsymbol{z} := \phi(\boldsymbol{x}, \boldsymbol{y}) = vec\left(\sum_{r=1}^{R} (\boldsymbol{E}^r \boldsymbol{x}) \otimes (\boldsymbol{F}^r \boldsymbol{y})\right), \tag{5}$$

where $\boldsymbol{E}^r = [\boldsymbol{e}_1^r, \boldsymbol{e}_2^r, ..., \boldsymbol{e}_M^r]^T$ and $\boldsymbol{F}^r = [\boldsymbol{f}_1^r, \boldsymbol{f}_2^r, ..., \boldsymbol{f}_N^r]^T$ for $r = 1, 2, ..., R$. With such a low-rank assumption, we avoid computing the high-dimensional correlation matrix $\boldsymbol{x} \otimes \boldsymbol{y}$, which considerably reduces the model parameters from $D_X D_Y D_Z$ to $R(MD_X + ND_Y)$ with a small value of R.

A similar low-rank assumption is also used in Ben-Younes et al. (2017) and Kim et al. (2016), in which the two input feature vectors are first projected to a common vector space and then fused via the Hadamard product. Assuming the input feature vectors are of the same dimension D and the output feature vector is of dimension d, then such operation requires $\mathcal{O}(Dd + d)$ operations to compute and requires $\mathcal{O}(Dd)$ memory to store. In contrast, our method requires $\mathcal{O}(D\sqrt{d} + d)$ for computation and $\mathcal{O}(D\sqrt{d})$ for storage. Since in practice it normally requires more dimensions to represent a higher-order feature, i.e. $d \gg D$, our method has a consistently better runtime (see Tab. 1) and hence is more suitable to be employed in a sophisticated deep neural network.

3.2 Random projection

To compensate for the loss in model capacity caused by the low-rank assumption, we observe that the entries and associated distributions of $\boldsymbol{E}$ and $\boldsymbol{F}$ defined in Eq. (4) and Eq. (5) can indeed influence the model capacity, without adding or removing learnable parameters, network layers, etc.

Inspired by this observation, we propose to manipulate the model capacity by randomly sampling the entries of $\boldsymbol{E}$ and $\boldsymbol{F}$ from specific distributions and then perform random projection. Unlike an end-to-end learning via back-propagation, our approach provides an alternative way of building expressive representation that is explainable and is simple to use in practice.

Rademacher random projection. Motivated by Kar & Karnick (2012) and Gao et al. (2016), we specify model parameters, i.e. the projection matrices $\boldsymbol{E}^r$ and $\boldsymbol{F}^r$ in the bilinear model (4) or (5), with random samples from the Rademacher distribution. Below we show that the bilinear model given in Eq. (4) *unbiasedly* approximates, with high probability, a feature map to a reproducing kernel Hilbert space (RKHS), in which the associated kernel is the multiplication of two linear kernels in $\mathcal{X}$ and $\mathcal{Y}$, respectively.

Theorem 1. *Let $\boldsymbol{E}^r \in \mathbb{R}^{M \times D_X}$ and $\boldsymbol{F}^r \in \mathbb{R}^{N \times D_Y}$ for any $r \in \{1, 2, ..., R\}$ be Rademacher random matrices whose entries are determined by an independent Rademacher random variable $\sigma \in \{-1, 1\}$. For any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ and $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathcal{Y}$, let $\boldsymbol{z}_1 = \phi(\boldsymbol{x}_1, \boldsymbol{y}_1)$ and $\boldsymbol{z}_2 = \phi(\boldsymbol{x}_2, \boldsymbol{y}_2)$ be the output features given by Eq. (4). Define a kernel function by $k(\boldsymbol{z}_1, \boldsymbol{z}_2) = \langle \boldsymbol{z}_1, \boldsymbol{z}_2 \rangle$, then we have*

$$\mathbb{E}[k(\boldsymbol{z}_1, \boldsymbol{z}_2)] = RMN \langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \langle \boldsymbol{y}_1, \boldsymbol{y}_2 \rangle.$$

Next, we characterize the error of such kernel approximations.

Corollary 1. *Let $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ be defined as in Theorem 1. Let $k(\boldsymbol{z}_1, \boldsymbol{z}_2) = \frac{1}{R}\langle \boldsymbol{z}_1, \boldsymbol{z}_2 \rangle$. Then, the following inequality holds:*

$$\mathbb{P}\left(|k(\boldsymbol{z}_1, \boldsymbol{z}_2) - \mathbb{E}[k(\boldsymbol{z}_1, \boldsymbol{z}_2)]| > \epsilon\right) \leq 2 \exp\left(-\frac{\epsilon^2 MN}{2p^8 \tilde{R}^8}\right), \tag{6}$$

for some constants $\epsilon > 0$, and $p \geq 1$, $\tilde{R} \geq 1$, which are independent of the feature dimensions.

Proofs of both results can be found in Appendix A and B. More details on the constants p and $\tilde{R}$ can be found in Kar & Karnick (2012). To remove the effect of the scaling factors, we rewrite Eq. (5) as

$$\boldsymbol{z} = \phi(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{R\sqrt{MN}} \cdot vec\left(\sum_{r=1}^{R} (\boldsymbol{E}^r \boldsymbol{x}) \otimes (\boldsymbol{F}^r \boldsymbol{y})\right). \tag{7}$$

Therefore, Eq. (7) with *binary tensor entries* is capable of capturing second-order interactions between two features. We refer this bilinear form as “RPBinary” in the experiment section.

Gaussian random projection. To increase the model capacity, a common approach is to apply the nonlinear activation function $\varphi(\cdot)$, which gives

$$\boldsymbol{z} := \phi(\boldsymbol{x}, \boldsymbol{y}) = vec\left(\sum_{r=1}^{R} \varphi(\boldsymbol{E}^r \boldsymbol{x}) \otimes \varphi(\boldsymbol{F}^r \boldsymbol{y})\right). \tag{8}$$

Inspired by Yu et al. (2016), we consider

$$\boldsymbol{E}^r = \frac{1}{\sigma^r}\boldsymbol{I}_{M\times D_X}\boldsymbol{R}^r\boldsymbol{P}^r, \quad \boldsymbol{F}^r = \frac{1}{\rho^r}\boldsymbol{I}_{N\times D_Y}\boldsymbol{S}^r\boldsymbol{Q}^r \quad \text{with} \quad r = 1, 2, ..., R, \tag{9}$$

where $\boldsymbol{R}^r$ and $\boldsymbol{S}^r$ are diagonal matrices with diagonal entries sampled i.i.d. from the chi-squared distributions $\chi^2(D_X)$ and $\chi^2(D_Y)$ with D_X and D_Y degrees-of-freedom, respectively, $\boldsymbol{P}^r$ and $\boldsymbol{Q}^r$ are uniformly distributed random orthogonal matrices[2], and $\boldsymbol{I}_{M\times D_X}$ and $\boldsymbol{I}_{N\times D_Y}$ are identity matrices with the first M and N rows, respectively. Here, $\{\sigma^r\}_{r=1}^R$ and $\{\rho^r\}_{r=1}^R$ are tunable bandwidth parameters.

When the nonlinear function in Eq. (8) is given by $\varphi(\boldsymbol{Ex}) := \sqrt{1/M}[\sin(\boldsymbol{Ex}), \cos(\boldsymbol{Ex})]$ and $\varphi(\boldsymbol{Fy}) := \sqrt{1/N}[\sin(\boldsymbol{Fy}), \cos(\boldsymbol{Fy})]$, the resulting representation approximates feature maps to the RKHSs corresponding to a composition of Gaussian kernels (see Appendix C for the proof).

Theorem 2. *Let $\boldsymbol{E}^r \in \mathbb{R}^{M\times D_X}$ and $\boldsymbol{F}^r \in \mathbb{R}^{N\times D_Y}$ for any $r \in \{1, 2, ..., R\}$ be random matrices whose entries are determined as in Eq. (9). For any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ and $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathcal{Y}$, let $\boldsymbol{z}_1 = \phi(\boldsymbol{x}_1, \boldsymbol{y}_1)$ and $\boldsymbol{z}_2 = \phi(\boldsymbol{x}_2, \boldsymbol{y}_2)$ be the output features in Eq. (8). Define a kernel function $k(\boldsymbol{z}_1, \boldsymbol{z}_2) = \langle \boldsymbol{z}_1, \boldsymbol{z}_2\rangle$, then we have*

$$\mathbb{E}[k(\boldsymbol{z}_1, \boldsymbol{z}_2)] = \sum_{r=1}^{R} \exp\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2}{2\sigma_r^2}\right)\exp\left(-\frac{\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2}{2\rho_r^2}\right) + b, \tag{10}$$

where $b := \sum_{r=1}^{R}\sum_{r'=r+1}^{R} \mathbb{E}[\langle\varphi(\boldsymbol{E}^r\boldsymbol{x}_1), \varphi(\boldsymbol{E}^{r'}\boldsymbol{x}_2)\rangle]\mathbb{E}[\langle\varphi(\boldsymbol{F}^r\boldsymbol{y}_1), \varphi(\boldsymbol{F}^{r'}\boldsymbol{y}_2)\rangle]$.

A characterization of the variance of $k(\boldsymbol{z}_1, \boldsymbol{z}_2)$ in Theorem 2 is given in Appendix D, which suggests that higher output feature dimension can reduce the variance of $k(\boldsymbol{z}_1, \boldsymbol{z}_2)$. Consequently, just by using the periodic function $\varphi(\cdot)$ and the Gaussian random projection, we can obtain infinite-order information without extra computational cost. We refer this form as "RPGaussianFull" in the experiment section. In principal, one can extract different types of high-order information by using different nonlinear functions in Eq. (8). Further investigations on such problems are currently beyond our scope.

In practice, when used as an intermediate layer in deep neural networks, $\sin$ and $\cos$ functions are known to be difficult and unreliable to train using back-propagation (Parascandolo et al., 2016). Therefore, we perform Taylor expansion of $\sin$ and $\cos$, and only use the first term. Namely, we approximate $\sin(\boldsymbol{E}x) \approx \boldsymbol{Ex}$ and $\cos(\boldsymbol{E}x) \approx 1$. Then, we can discard the nonlinear function $\varphi(\cdot)$ in Eq. (8), and also can apply scaling factors as in Eq. (7). We refer this approximated version as "RPGaussian" in our experiments. In addition, we adopt a simpler version $\boldsymbol{E}^r = \frac{\sqrt{D_X}}{\sigma^r}\boldsymbol{I}_{\boldsymbol{M}\times\boldsymbol{D_X}}\boldsymbol{P}^r$ and $\boldsymbol{F}^r = \frac{\sqrt{D_Y}}{\rho^r}\boldsymbol{I}_{\boldsymbol{N}\times\boldsymbol{D_Y}}\boldsymbol{Q}^r$. According to Yu et al. (2016), such a simplified version exhibits similar empirical behavior to the original version, especially when the feature dimensionality is high. Moreover, rather than regarding the Gaussian radii as hyper-parameters, we learn them via back-propagation when employing the model (7) as an intermediate layer in a deep neural network.

Last but not least, it is instructive to note that, in addition to what we have proposed, the distributions of $\boldsymbol{E}$ and $\boldsymbol{F}$ can be arbitrary. We can even model these distributions using deep generative models, which is the subject of our future work.

4 Experiments

We conduct experiments for the task of fine-grained temporal action segmentation, which aims at assigning each individual frame an action label. In such experiments, the two input features $\boldsymbol{x}$ and

[2]Specifically, $\boldsymbol{P}^r$ and $\boldsymbol{Q}^r$ are uniformly distributed on the Stiefel manifold (Yu et al., 2016; Muirhead, 2009).

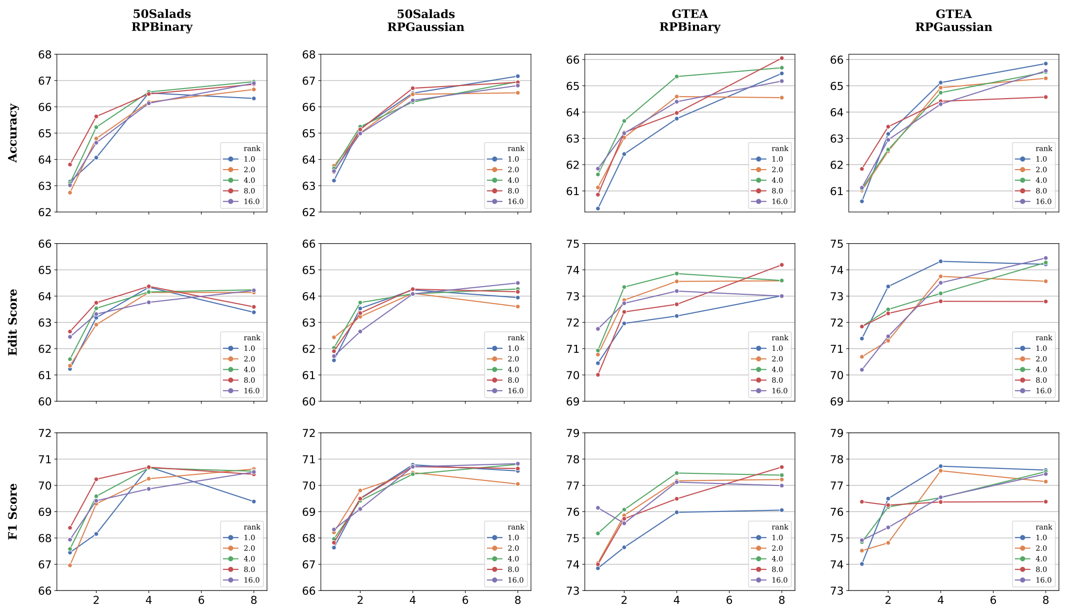

Figure 1: Performance of our RPBinary and RPGaussian model, versus dimension / rank, on datasets **50Salads** and **GTEA**. In each plot, x-axis is the multiplier on the number of matrix rows N, y-axis is the respective performance measure, and colors denote different ranks.

$\boldsymbol{y}$ in Eq. (7) and Eq. (8) are identical, and we set the same number of rows (i.e. $M = N$) to matrices $\boldsymbol{E}^r$ and $\boldsymbol{F}^r$ for all $r = 1, \ldots, R$. Consequently, there only remain two hyper-parameters in the bilinear model, i.e. the rank R and the number of rows N of matrices. The objectives of our experiments are two-fold: (i) To verify the effectiveness of our method, we adopt the temporal convolutional net (TCN) (Lea et al., 2017) due to its simple structure and replace the max pooling in TCN by bilinear pooling as in Zhang et al. (2019). (ii) To demonstrate how our method can be used in practice, we propose a bilinear residual module to merge the first and second-order information, and combine it with the multi-stage temporal convolutional net (MS-TCN) (Farha & Gall, 2019) to yield state-of-the-art performance.

Datasets and evaluation metrics. We evaluate our method on the **50Salads** dataset (Stein & McKenna, 2013) and the **GTEA** dataset (Fathi et al., 2011; Li et al., 2015). For a fair comparison, in our experiments with TCN, we use the identical frame-wise features and temporal resolutions as Lea et al. (2017). Also, in our experiments with MS-TCN, we use the identical frame-wise features and temporal resolutions as Farha & Gall (2019). To evaluate the performance, we use three standard metrics, i.e. frame-wise accuracy, edit score and F1 score as in Lea et al. (2017) and Farha & Gall (2019). For the F1 score, we consider the intersection-over-union (IoU) ratio of 0.1, 0.25 and 0.5, and denote them as F1@0.1, F1@0.25 and F1@0.5, respectively. Without explicit mentioning, our F1 score means F1@0.1. Detailed definitions of these metrics are explained in Lea et al. (2017) and Farha & Gall (2019). Since each dataset has several splits, we report the results of cross-validation.

4.1 Action segmentation with TCN architecture

We use the default architecture of TCN (Lea et al., 2017), which comprises an encoder and a decoder with symmetric modules. The convolutional layer in each individual encoder has 64 and 96 filters, respectively. We train the model using the Adam optimizer (Kingma & Ba, 2014) with a fixed learning rate of 10^{-4}. Batch size is set to 8, and the training process terminates after 300 epochs.

Ablation study: Comparison between bilinear forms. In this experiment, we set the rank $R = 1$ and $N = D/2$ for our methods where D is the input feature dimension to our bilinear model. The results are shown in the first part of Tab. 1. For the performance evaluation, we repeat the experiment 3 times and report the result with the highest sum of the three metrics. To evaluate the efficiency, we report the runtime per batch (batch size=8) which is the averaged result after training

Table 1: Comparison with different bilinear pooling methods in terms of *accuracy/edit score/F1 score* and runtime (millisecond). For each metric and each setting, the best result is in boldface. "LearnableProjection" indicates $\boldsymbol{E}$ and $\boldsymbol{F}$ in Eq. (7) are learned via back-propagation. "RPBinary" indicates the model Eq. (7) with Rademacher random projection. "RPGaussianFull" and "RPGaussian" indicate the model Eq. (8) with and without $\varphi(\cdot)$, respectively, in which Gaussian random projection is employed. In the column of complexity, D and d denote the input and output feature dimension of our bilinear model, respectively.

		50Salads		**GTEA**	
Method	Complexity	Runtime	Performance	Runtime	Performance
Ours (RPBinary)	$\mathcal{O}(D\sqrt{d}+d)$	**68.3**	66.0/**65.9**/70.9	80.1	65.2/73.2/77.0
Ours (RPGaussian)	$\mathcal{O}(D\sqrt{d}+d)$	68.9	67.6/65.2/**72.9**	**69.9**	**66.9**/76.5/**79.8**
Ours (RPGaussianFull)	$\mathcal{O}(D\sqrt{d}+d)$	73.7	64.1/63.4/69.6	70.0	64.5/73.0/78.7
Ours (LearnableProjection)	$\mathcal{O}(D\sqrt{d}+d)$	78.6	66.4/65.0/70.5	81.6	64.8/74.0/77.5
Compact (Gao et al., 2016)	$\mathcal{O}(D+d\log d)$	95.9	67.2/65.8/71.7	120.2	65.9/75.3/78.1
Hadamard (Kim et al., 2016)	$\mathcal{O}(Dd+d)$	83.8	**67.7**/64.4/71.5	83.6	66.0/**76.6**/79.0
FBP (Li et al., 2017b)	$\mathcal{O}((2k+1)Dd+d)$	130.5	64.0/61.0/67.5	127.6	63.4/71.6/74.1

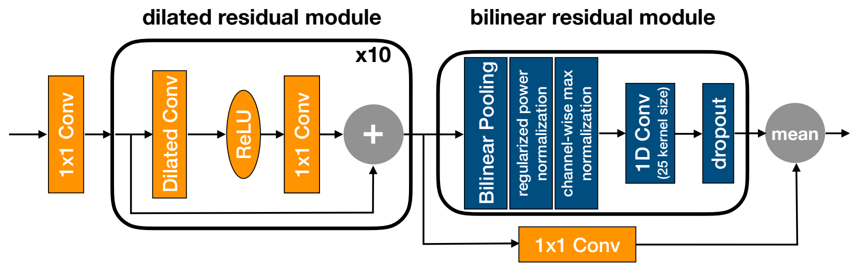

Figure 2: A combination of MS-TCN Farha & Gall (2019) and our bilinear pooling method, in which the blue layers are proposed by us and the orange layers are proposed by MS-TCN.

with the first split for 300 epochs for each dataset. Overall, the results suggest that "RPGaussian" and "RPBinary" have comparable performances, and outperform "LearnableProjection" and "RPGaussianFull". While "LearnableProjection" is more flexible, it might require a more sophisticated training to achieve the same performance as our random projection methods. Also, we suspect that the inferior performance of "RPGaussianFull" is due to the difficulty of training.

In addition, we compare our methods with three widely used bilinear pooling methods, i.e., **Compact** (Gao et al., 2016), **Hadamard** (Kim et al., 2016) and **FBP** (Li et al., 2017b). We use the same output feature dimension for fair comparison. The results are shown in the second part of Tab. 1. They suggest that these three methods perform inferior or comparably in terms of the three metrics of action parsing, and are clearly less efficient, than our methods.

Ablation study: Investigation on the hyper-parameters of our bilinear forms. Here we investigate the influence of the rank and the output feature dimension of RPGaussian and RPBinary, with the bilinear model in Eq. (7). Fig. 1 shows the dependence of the model performance on ranks $R \in \{1, 2, 4, 8, 16\}$ and matrix rows $N \in \{1, 2, 4, 8\} \times [\sqrt{D}]$ where $[n]$ denotes the nearest integer to n. In this case, the output feature dimension is then $\{1, 4, 16, 64\} \times D$. In all plots, the performance increases consistently with the matrix row N (hence the output feature dimension). This result is in line with our theoretical analysis on the kernel approximation error bound (see Corollary 1 and Appendix D): Larger values of M and N can yield lower variance upper bounds, hence better kernel approximation. One can also observe that the performance saturates when further increasing the output feature dimension. This is due to the limited capacity of the corresponding RKHS. Moreover, one can observe that increasing the rank may not consistently yield better performance. An optimal rank depends on the dataset and the applied bilinear model.

Table 2: Comparison with other models on temporal action segmentation task. The best results are in **boldface**, the second-best results are underlined.

	50 Salads					GTEA				
	Acc.	Edit	F1@0.1	F1@0.25	F1@0.5	Acc.	Edit	F1@0.1	F1@0.25	F1@0.5
Ours (RPBinary)	79.9	70.7	78.0	75.2	65.4	77.0	81.4	86.5	**84.5**	71.7
Ours (RPGaussian)	80.6	**71.0**	**78.4**	**75.8**	66.7	**77.2**	82.2	86.7	84.3	**72.7**
Ours (RPGaussianFull)	77.4	67.5	74.9	70.9	60.7	76.8	**82.5**	**87.3**	84.3	71.8
TCN (Lea et al., 2017)	64.7	59.8	68.0	63.9	52.6	64.0	-	72.2	69.3	56.0
TDRN (Lei & Todorovic, 2018)	68.1	66.0	72.9	68.5	57.2	70.1	74.1	79.2	74.4	62.7
MS-TCN (Farha & Gall, 2019)	80.7	67.9	76.3	74.0	64.5	76.3	79.0	85.8	83.4	69.8
MS-TCN + Compact (Gao et al., 2016)	**81.1**	70.7	77.6	75.1	**67.1**	75.8	80.9	86.0	83.7	70.2
MS-TCN + Hadamard (Kim et al., 2016)	78.3	68.3	75.3	72.4	62.5	77.0	81.5	86.0	83.5	70.7
MS-TCN + FBP (Li et al., 2017b)	78.5	70.0	76.5	73.6	64.6	75.4	79.3	84.0	81.8	69.9

4.2 Action segmentation with MS-TCN

In this section, we demonstrate how to effectively incorporate our method into the state-of-the-art action parsing network, MS-TCN (Farha & Gall, 2019), to superior performance.

Implementation Details. Based on our previous experimental results, we set rank $R = 4$ and $N = D/2$ for our bilinear model. Furthermore, we propose a bilinear residual module to merge the first and the second-order information, as illustrated in Fig. 2. First, we use bilinear pooling to extract the second-order information, and then use a regularized power normalization (Zhang et al., 2019) to densify the feature and use channel-wise max normalization to re-scale the feature value (Lea et al., 2017). Afterwards, we use a convolution layer to reduce the feature dimension to the number of classes. Since the second-order information tends to partition an action into smaller segments (Zhang et al., 2019, Fig. 1), we use a larger convolution receptive field 25 according to Lea et al. (2017). To prevent overfitting we use a dropout layer, and then we compute the average between the first-order information and the second-order information.

Our bilinear residual module is applied at the end of each single stage of MS-TCN. To conduce fair comparison with the baseline MS-TCN model, we keep other model configurations and the loss function (including the hyper-parameters) unchanged. Similarly to Farha & Gall (2019), we use the Adam (Kingma & Ba, 2014) optimizer with a learning rate of 0.0005. The batch size is set to 1.

Result. We compare our method with several state-of-the-art methods on the action segmentation task. As shown in Tab. 2, our models (especially RPBinary and RPGaussian) consistently outperform the state-of-the-arts (TCN, TDRN, MS-TCN), validating the effectiveness of the proposed bilinear model. Note that, even with RPGaussianFull, which is hard to train by back-propagation, our bilinear form achieves competitive performance on the GTEA dataset. For a fair and complete comparison, we further integrate three widely used light-weight bilinear models into the MS-TCN model, namely, **Compact** (Gao et al., 2016), **Hadamard** (Kim et al., 2016) and **FBP** (Li et al., 2017b). Again, the proposed bilinear models show superior performance on most of the evaluation metrics. Together with the results presented in Tab. 1, they clearly demonstrate the representational power and the computational efficiency of the proposed bilinear models.

5 Conclusion

In this work, we propose a novel bilinear model for fusing high-dimensional features. To reduce the number of model parameters, we utilize low-rank tensor decomposition. Instead of using element-wise product as in other works, we use the outer product of the features to model the high-order correlations among feature channels. To enrich the model representiveness while retaining the number of parameters, we use random projection to approximate feature maps to reproducing kernel Hilbert spaces associated with kernel compositions. To validate the effectiveness of our method, we perform extensive experiments on the action segmentation task, and have achieved state-of-the-art performance on challenging benchmarks. Our bilinear pooling operation is lightweight, easy to use, and can serve as a natural tool for fine-grained visual understanding and information fusion. In the future we will investigate how to combine our work with the deep generative models. That is, instead of sampling entries from a pre-defined distribution, we aim to learn to model these distributions using deep generative models.

REFERENCES

Richard E Bellman. *Adaptive control processes: a guided tour*, volume 2045. Princeton university press, 2015.

Hedi Ben-Younes, Rémi Cadene, Matthieu Cord, and Nicolas Thome. Mutan: Multimodal tucker fusion for visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pp. 2612–2620, 2017.

Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *European Conference on Computer Vision*, pp. 430–443. Springer, 2012.

Anoop Cherian, Piotr Koniusz, and Stephen Gould. Higher-order pooling of cnn features via kernel linearization for action recognition. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 130–138. IEEE, 2017.

Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge Belongie. Kernel pooling for convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2930, 2017.

Ali Diba, Vivek Sharma, and Luc Van Gool. Deep temporal linear encoding networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2329–2338, 2017.

Yazan Abu Farha and Juergen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Alireza Fathi, Xiaofeng Ren, and James M Rehg. Learning to recognize objects in egocentric activities. In *IEEE Conference On Computer Vision and Pattern Recognition (CVPR)*, pp. 3281–3288. IEEE, 2011.

Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1933–1941, 2016.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv:1606.01847*, 2016.

Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 317–326, 2016.

Rohit Girdhar and Deva Ramanan. Attentional pooling for action recognition. In *Advances in Neural Information Processing Systems*, pp. 34–45, 2017.

Mengran Gou, Fei Xiong, Octavia Camps, and Mario Sznaier. Monet: Moments embedding network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3175–3183, 2018.

Purushottam Kar and Harish Karnick. Random feature maps for dot product kernels. In *Artificial Intelligence and Statistics*, pp. 583–591, 2012.

Jin-Hwa Kim, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Hadamard product for low-rank bilinear pooling. *arXiv preprint arXiv:1610.04325*, 2016.

Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear Attention Networks. In *Advances in Neural Information Processing Systems 31*, pp. 1571–1581, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3): 455–500, 2009.

Shu Kong and Charless Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7025–7034. IEEE, 2017.

Piotr Koniusz, Fei Yan, Philippe-Henri Gosselin, and Krystian Mikolajczyk. Higher-order occurrence pooling for bags-of-words: Visual concept detection. *IEEE transactions on pattern analysis and machine intelligence*, 39(2):313–326, 2017.

Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1003–1012, July 2017.

Peng Lei and Sinisa Todorovic. Temporal deformable residual networks for action segmentation in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6742–6751, 2018.

Peihua Li, Jiangtao Xie, Qilong Wang, and Wangmeng Zuo. Is second-order information helpful for large-scale visual recognition. In *IEEE international conference on computer vision (ICCV). IEEE*, pp. 2070–2078, 2017a.

Peihua Li, Jiangtao Xie, Qilong Wang, and Zilin Gao. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 947–955, 2018.

Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Factorized bilinear models for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2079–2087, 2017b.

Yin Li, Zhefan Ye, and James M Rehg. Delving into egocentric actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 287–295, 2015.

Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pp. 1449–1457, 2015.

Tsung-Yu Lin, Subhransu Maji, and Piotr Koniusz. Second-order democratic aggregation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 620–636, 2018a.

Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear convolutional neural networks for fine-grained visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1309–1322, 2018b.

Robb J Muirhead. *Aspects of multivariate statistical theory*, volume 197. John Wiley & Sons, 2009.

Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Taming the waves: sine as activation function in deep neural networks. 2016.

Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 239–247. ACM, 2013.

Sebastian Stein and Stephen J McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pp. 729–738. ACM, 2013.

Qilong Wang, Peihua Li, and Lei Zhang. G2denet: Global gaussian distribution embedding network and its application to visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2730–2739, 2017.

Xing Wei, Yue Zhang, Yihong Gong, Jiawei Zhang, and Nanning Zheng. Grassmann pooling as compact homogeneous bilinear pooling for fine-grained visual classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 355–370, 2018.

Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In *Advances in Neural Information Processing Systems*, pp. 1975–1983, 2016.

Kaicheng Yu and Mathieu Salzmann. Statistically-motivated second-order pooling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 600–616, 2018.

Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. *IEEE International Conference on Computer Vision (ICCV)*, pp. 1839–1848, 2017.

Zhou Yu, Jun Yu, Chenchao Xiang, Jianping Fan, and Dacheng Tao. Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering. *IEEE Transactions on Neural Networks and Learning Systems*, 29(12):5947–5959, 2018.

Yan Zhang, Siyu Tang, Krikamol Muandet, Christian Jarvers, and Heiko Neumann. Local temporal bilinear pooling for fine-grained action parsing. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2019. URL `https://arxiv.org/abs/1812.01922`.

APPENDIX

A PROOF OF THEOREM 1

Proof. It follows from Eq. (5) and the property of an inner product of rank-one operators that

$$\begin{aligned}
k(\boldsymbol{z}_1, \boldsymbol{z}_2) := \langle \boldsymbol{z}_1, \boldsymbol{z}_2 \rangle &= \left\langle \sum_{r=1}^{R} vec(\boldsymbol{E}^r \boldsymbol{x}_1 \otimes \boldsymbol{F}^r \boldsymbol{y}_1), \sum_{r=1}^{R} vec(\boldsymbol{E}^r \boldsymbol{x}_2 \otimes \boldsymbol{F}^r \boldsymbol{y}_2) \right\rangle_{\mathbb{R}^{MN}} \\
&= \sum_{r=1}^{R} \sum_{r'=1}^{R} \left\langle vec(\boldsymbol{E}^r \boldsymbol{x}_1 \otimes \boldsymbol{F}^r \boldsymbol{y}_1), vec(\boldsymbol{E}^{r'} \boldsymbol{x}_2 \otimes \boldsymbol{F}^{r'} \boldsymbol{y}_2) \right\rangle_{\mathbb{R}^{MN}} \\
&= \sum_{r=1}^{R} \sum_{r'=1}^{R} \left\langle \boldsymbol{E}^r \boldsymbol{x}_1, \boldsymbol{E}^{r'} \boldsymbol{x}_2 \right\rangle \left\langle \boldsymbol{F}^r \boldsymbol{y}_1, \boldsymbol{F}^{r'} \boldsymbol{y}_2 \right\rangle \\
&= \sum_{r=1}^{R} \sum_{r'=1}^{R} \left(\sum_{i=1}^{M} \langle \boldsymbol{e}_i^r, \boldsymbol{x}_1 \rangle \langle \boldsymbol{e}_i^{r'}, \boldsymbol{x}_2 \rangle \right) \left(\sum_{j=1}^{N} \langle \boldsymbol{f}_j^r, \boldsymbol{y}_1 \rangle \langle \boldsymbol{f}_j^{r'}, \boldsymbol{y}_2 \rangle \right). \qquad (11)
\end{aligned}$$

Then, it follows that

$$\begin{aligned}
\mathbb{E}[k(\boldsymbol{z}_1, \boldsymbol{z}_2)] &= \sum_{r=1}^{R} \sum_{r'=1}^{R} \left(\sum_{i=1}^{M} \mathbb{E}[\langle \boldsymbol{e}_i^r, \boldsymbol{x}_1 \rangle \langle \boldsymbol{e}_i^{r'}, \boldsymbol{x}_2 \rangle] \right) \left(\sum_{j=1}^{N} \mathbb{E}[\langle \boldsymbol{f}_j^r, \boldsymbol{y}_1 \rangle \langle \boldsymbol{f}_j^{r'}, \boldsymbol{y}_2 \rangle] \right) \\
&= \sum_{r=1}^{R} \left(\sum_{i=1}^{M} \mathbb{E}[\langle \boldsymbol{e}_i^r, \boldsymbol{x}_1 \rangle \langle \boldsymbol{e}_i^{r}, \boldsymbol{x}_2 \rangle] \right) \left(\sum_{j=1}^{N} \mathbb{E}[\langle \boldsymbol{f}_j^r, \boldsymbol{y}_1 \rangle \langle \boldsymbol{f}_j^{r}, \boldsymbol{y}_2 \rangle] \right) \\
&\quad + \sum_{r=1}^{R} \sum_{r'=r+1}^{R} \left(\sum_{i=1}^{M} \mathbb{E}[\langle \boldsymbol{e}_i^r, \boldsymbol{x}_1 \rangle \langle \boldsymbol{e}_i^{r'}, \boldsymbol{x}_2 \rangle] \right) \left(\sum_{j=1}^{N} \mathbb{E}[\langle \boldsymbol{f}_j^r, \boldsymbol{y}_1 \rangle \langle \boldsymbol{f}_j^{r'}, \boldsymbol{y}_2 \rangle] \right) \\
&= RMN \langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \langle \boldsymbol{y}_1, \boldsymbol{y}_2 \rangle.
\end{aligned}$$

The last equation follows from Kar & Karnick (2012, Lemma 2) and the fact that $\boldsymbol{e}_i^r$ and $\boldsymbol{f}_j^r$ are zero-mean random variables for all $i = 1, \ldots, M$, $j = 1, \ldots, N$ and $r = 1, \ldots, R$. □

B PROOF OF COROLLARY 1

Proof. Let $W_{ij} := \langle \boldsymbol{e}_i, \boldsymbol{x}_1 \rangle \langle \boldsymbol{e}_i, \boldsymbol{x}_2 \rangle \langle \boldsymbol{f}_j, \boldsymbol{y}_1 \rangle \langle \boldsymbol{f}_j, \boldsymbol{y}_2 \rangle$ for $i = 1, \ldots, M$ and $j = 1, \ldots, N$. For each W_{ij}, it follows from Kar & Karnick (2012, Lemma 4) that

$$-p^2 f(p\tilde{R}^2)^2 \leq W_{ij} \leq p^2 f(p\tilde{R}^2)^2,$$

where we assume without loss of generality that $p \geq 1$ and $\tilde{R} \geq 1$. In our case, $f(x) = x$. Then, we have $-p^4\tilde{R}^4 \leq W_{ij} \leq p^4\tilde{R}^4$. Let $S_{MN} := \frac{1}{R}\sum_{i=1}^{M}\sum_{j=1}^{N} W_{ij}$. Hence, we have $\mathbb{E}[S_{MN}] = MN\langle \boldsymbol{x}_1, \boldsymbol{x}_2\rangle\langle \boldsymbol{y}_1, \boldsymbol{y}_2\rangle$. Then, it follows from Hoeffding's inequality that, for all $\epsilon > 0$,

$$\mathbb{P}(|S_{MN} - \mathbb{E}[S_{MN}]| \geq \epsilon) \leq 2\exp\left(-\frac{\epsilon^2 MN}{2p^8\tilde{R}^8}\right). \tag{12}$$

This concludes the proof. $\square$

C Proof of Theorem 2

Proof. Let $R = 1$. Then, $k(\boldsymbol{z}_1, \boldsymbol{z}_2) := \langle \boldsymbol{z}_1, \boldsymbol{z}_2\rangle = \langle \varphi(\boldsymbol{E}\boldsymbol{x}_1), \varphi(\boldsymbol{E}\boldsymbol{x}_2)\rangle\langle \varphi(\boldsymbol{F}\boldsymbol{x}_1), \varphi(\boldsymbol{F}\boldsymbol{x}_2)\rangle$ where

$$\begin{aligned}\varphi(\boldsymbol{E}\boldsymbol{x}) &:= \frac{1}{\sqrt{M}}[\sin(\boldsymbol{e}_1^\top\boldsymbol{x}), \ldots, \sin(\boldsymbol{e}_M^\top\boldsymbol{x}), \cos(\boldsymbol{e}_1^\top\boldsymbol{x}), \ldots, \cos(\boldsymbol{e}_M^\top\boldsymbol{x})],\\ \varphi(\boldsymbol{F}\boldsymbol{y}) &:= \frac{1}{\sqrt{N}}[\sin(\boldsymbol{f}_1^\top\boldsymbol{y}), \ldots, \sin(\boldsymbol{f}_M^\top\boldsymbol{y}), \cos(\boldsymbol{f}_1^\top\boldsymbol{y}), \ldots, \cos(\boldsymbol{f}_M^\top\boldsymbol{y})].\end{aligned}$$

With $\boldsymbol{E} = \frac{1}{\sigma}\boldsymbol{I}_{M\times D_X}\boldsymbol{R}\boldsymbol{P}$ and $\boldsymbol{F} = \frac{1}{\rho}\boldsymbol{I}_{N\times D_Y}\boldsymbol{S}\boldsymbol{Q}$, we have

$$\begin{aligned}\mathbb{E}[k(\boldsymbol{z}_1, \boldsymbol{z}_2)] &= \mathbb{E}[\langle \varphi(\boldsymbol{E}\boldsymbol{x}_1), \varphi(\boldsymbol{E}\boldsymbol{x}_2)\rangle]\mathbb{E}[\langle \varphi(\boldsymbol{F}\boldsymbol{y}_1), \varphi(\boldsymbol{F}\boldsymbol{y}_2)\rangle]\\ &= \exp\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2}{2\sigma^2}\right)\exp\left(-\frac{\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2}{2\rho^2}\right),\end{aligned}$$

where the last equality follows from Yu et al. (2016, Theorem 1). For $R > 1$, we have

$$k(\boldsymbol{z}_1, \boldsymbol{z}_2) := \langle \boldsymbol{z}_1, \boldsymbol{z}_2\rangle = \sum_{r=1}^{R}\sum_{r'=1}^{R}\langle \varphi(\boldsymbol{E}^r\boldsymbol{x}_1), \varphi(\boldsymbol{E}^{r'}\boldsymbol{x}_2)\rangle\langle \varphi(\boldsymbol{F}^r\boldsymbol{x}_1), \varphi(\boldsymbol{F}^{r'}\boldsymbol{x}_2)\rangle.$$

Hence, with $\boldsymbol{E}^r = \frac{1}{\sigma^r}\boldsymbol{I}_{M\times D_X}\boldsymbol{R}^r\boldsymbol{P}^r$ and $\boldsymbol{F}^r = \frac{1}{\rho^r}\boldsymbol{I}_{N\times D_Y}\boldsymbol{S}^r\boldsymbol{Q}^r$,

$$\begin{aligned}\mathbb{E}[k(\boldsymbol{z}_1, \boldsymbol{z}_2)] &= \sum_{r=1}^{R}\sum_{r'=1}^{R}\mathbb{E}[\langle \varphi(\boldsymbol{E}^r\boldsymbol{x}_1), \varphi(\boldsymbol{E}^{r'}\boldsymbol{x}_2)\rangle]\mathbb{E}[\langle \varphi(\boldsymbol{F}^r\boldsymbol{y}_1), \varphi(\boldsymbol{F}^{r'}\boldsymbol{y}_2)\rangle]\\ &= \sum_{r=1}^{R}\mathbb{E}[\langle \varphi(\boldsymbol{E}^r\boldsymbol{x}_1), \varphi(\boldsymbol{E}^r\boldsymbol{x}_2)\rangle]\mathbb{E}[\langle \varphi(\boldsymbol{F}^r\boldsymbol{y}_1), \varphi(\boldsymbol{F}^r\boldsymbol{y}_2)\rangle]\\ &\quad+\sum_{r=1}^{R}\sum_{r'=r+1}^{R}\mathbb{E}[\langle \varphi(\boldsymbol{E}^r\boldsymbol{x}_1), \varphi(\boldsymbol{E}^{r'}\boldsymbol{x}_2)\rangle]\mathbb{E}[\langle \varphi(\boldsymbol{F}^r\boldsymbol{y}_1), \varphi(\boldsymbol{F}^{r'}\boldsymbol{y}_2)\rangle]\\ &= \sum_{r=1}^{R}\exp\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2}{2\sigma_r^2}\right)\exp\left(-\frac{\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2}{2\rho_r^2}\right)\\ &\quad+\sum_{r=1}^{R}\sum_{r'=r+1}^{R}\mathbb{E}[\langle \varphi(\boldsymbol{E}^r\boldsymbol{x}_1), \varphi(\boldsymbol{E}^{r'}\boldsymbol{x}_2)\rangle]\mathbb{E}[\langle \varphi(\boldsymbol{F}^r\boldsymbol{y}_1), \varphi(\boldsymbol{F}^{r'}\boldsymbol{y}_2)\rangle]\\ &= \sum_{r=1}^{R}\exp\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2}{2\sigma_r^2}\right)\exp\left(-\frac{\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2}{2\rho_r^2}\right) + b\end{aligned}$$

where $b := \sum_{r=1}^{R}\sum_{r'=r+1}^{R}\mathbb{E}[\langle \varphi(\boldsymbol{E}^r\boldsymbol{x}_1), \varphi(\boldsymbol{E}^{r'}\boldsymbol{x}_2)\rangle]\mathbb{E}[\langle \varphi(\boldsymbol{F}^r\boldsymbol{y}_1), \varphi(\boldsymbol{F}^{r'}\boldsymbol{y}_2)\rangle]$. $\square$

D Approximation error bound for Gaussian random projection

We characterize the variance of $k(\boldsymbol{z}_1, \boldsymbol{z}_2)$ used in Theorem 2. To simplify the presentation, we focus on the case that $R = 1$.

Corollary 2. *Let $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, and $k(\boldsymbol{z}_1, \boldsymbol{z}_2)$ be defined as in Theorem 2, as well as $R = 1$, $a = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2/\sigma$ and $b = \|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2/\rho$. Then, there exist functions f and g such that*

$$\mathrm{Var}(k(\boldsymbol{z}_1, \boldsymbol{z}_2)) \leq A \cdot B + A \cdot C + B \cdot D, \tag{13}$$

where

$$A = \frac{1}{2M}\left[\left(\left(1 - e^{-a^2}\right)^2 - \frac{M-1}{D_X} e^{-a^2} a^4\right) + \frac{f(a)}{D_X^2}\right], \qquad C = \left[\exp\left(-\frac{b^2}{2}\right)\right]^2$$
$$B = \frac{1}{2N}\left[\left(\left(1 - e^{-b^2}\right)^2 - \frac{N-1}{D_Y} e^{-b^2} b^4\right) + \frac{g(b)}{D_Y^2}\right], \qquad D = \left[\exp\left(-\frac{a^2}{2}\right)\right]^2.$$

Proof. Let R=1. Then, we have

$$k(\boldsymbol{z}_1, \boldsymbol{z}_2) = \underbrace{\langle \varphi(\boldsymbol{E}\boldsymbol{x}_1), \varphi(\boldsymbol{E}\boldsymbol{x}_2)\rangle}_{U} \underbrace{\langle \varphi(\boldsymbol{F}\boldsymbol{x}_1), \varphi(\boldsymbol{F}\boldsymbol{x}_2)\rangle}_{V}.$$

Since U and V are independent, we have

$$\mathrm{Var}(k(\boldsymbol{z}_1, \boldsymbol{z}_2)) = \mathrm{Var}(U)\mathrm{Var}(V) + \mathrm{Var}(U)(\mathbb{E}[V])^2 + \mathrm{Var}(V)(\mathbb{E}[U])^2, \tag{14}$$

where

$$(\mathbb{E}[U])^2 = (\mathbb{E}\left[\langle \varphi(\boldsymbol{E}\boldsymbol{x}_1), \varphi(\boldsymbol{E}\boldsymbol{x}_2)\rangle\right])^2 \qquad (\mathbb{E}[V])^2 = (\mathbb{E}\left[\langle \varphi(\boldsymbol{F}\boldsymbol{y}_1), \varphi(\boldsymbol{F}\boldsymbol{y}_2)\rangle\right])^2$$
$$\mathrm{Var}(U) = \mathrm{Var}\left(\langle \varphi(\boldsymbol{E}\boldsymbol{x}_1), \varphi(\boldsymbol{E}\boldsymbol{x}_2)\rangle\right) \qquad \mathrm{Var}(V) = \mathrm{Var}\left(\langle \varphi(\boldsymbol{F}\boldsymbol{y}_1), \varphi(\boldsymbol{F}\boldsymbol{y}_2)\rangle\right)$$

It follows from Yu et al. (2016, Theorem 1) that

$$(\mathbb{E}[U])^2 = \left(\exp\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2}{2\sigma^2}\right)\right)^2, \quad (\mathbb{E}[V])^2 = \left(\exp\left(-\frac{\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2}{2\rho^2}\right)\right)^2.$$

Let $a = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2/\sigma$ and $b = \|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2/\rho$. Then, by Yu et al. (2016, Theorem 1), there exists a function f and g such that

$$\begin{aligned} \mathrm{Var}(U) &\leq \frac{1}{2M}\left[\left(\left(1 - e^{-a^2}\right)^2 - \frac{M-1}{D_X} e^{-a^2} a^4\right) + \frac{f(a)}{D_X^2}\right] \\ \mathrm{Var}(V) &\leq \frac{1}{2N}\left[\left(\left(1 - e^{-b^2}\right)^2 - \frac{N-1}{D_Y} e^{-b^2} b^4\right) + \frac{g(b)}{D_Y^2}\right]. \end{aligned}$$

Substituting everything back into (14) yields the result. $\square$