Local Temporal Bilinear Pooling for Fine-Grained Action Parsing

Yan Zhang[1,2], Siyu Tang[2,3], Krikamol Muandet[2], Christian Jarvers[1], and Heiko Neumann[1]

[1]Institute of Neural Information Processing, Ulm University, Ulm, Germany
[2]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[3]University of Tübingen, Tübingen, Germany

Abstract

Fine-grained temporal action parsing is important in many applications, such as daily activity understanding, human motion analysis, surgical robotics and others requiring subtle and precise operations over a long-term period. In this paper we propose a novel bilinear pooling operation, which is used in intermediate layers of a temporal convolutional encoder-decoder net. In contrast to previous work, our proposed bilinear pooling is learnable and hence can capture more complex local statistics than the conventional counterpart. In addition, we introduce exact *lower-dimension representations of our bilinear forms, so that the dimensionality is reduced without suffering from information loss nor requiring extra computation. We perform extensive experiments to quantitatively analyze our model and show the superior performances to other state-of-the-art pooling work on various datasets.*

1 Introduction

Parsing fine-grained actions over time is important in many applications, which require understanding of subtle and precise operations over long-term periods, e.g. daily activities [1], surgical robots [2], human motion analysis [3] and animal behavior analysis in the lab [4]. Given a video or a generic time sequence of feature vectors, an action parsing algorithm aims at assigning each frame an action label, such that the entire sequence is partitioned into several disjoint semantic action primitives. Thus, tasks of action recognition, temporal semantic segmentation and action detection in untrimmed videos can be solved in one framework.

Recently, fine-grained action parsing algorithms based on deep convolutional nets are highly effective. For example, the method proposed in [5] and [6] first extracts frame-wise feature vectors via a spatial convolutional net, and then assigns action labels to individual frames via a temporal convolutional encoder-decoder (TCED) architecture. As reported, such TCED net outperforms other methods on challenging fine-grained action datasets of various scenarios.

While being straightforward, a notable caveat of the TCED architecture in [6] is that the max pooling operation embedded between convolutional layers in the encoder ignores high-order temporal structures, and hence cannot differentiate two fine-grained actions with identical first-order but different second-order statistics. Taking grasping object by hand as an example, when the feature vector of each frame is the concatenation of 3D positions of the finger tips, max pooling on several consecutive frames yields the hand position, and hence tells where to grasp the object. In parallel, the second-order information can indicate the finger scatter, and hence tells how to grasp the object. Thus, different orders of information are rather independent and complementary to precisely describe an action. Without the second-order information, it is hardly able to distinguish whether to grasp a coin or a book at the same position. Motivated by this example, as well as several recent studies showing that bilinear pooling outperforms first-order pooling on fine-grained tasks (e.g. [7–9]), we aim at introducing bilinear pooling into the TCED net, so that second-order statistics can be incorporated to produce better fine-grained action parsing results. We refer to Sec. 4.3 for detailed analysis of the benefits of second-order information.

However, combining such two methods is highly non-trivial, which requires to overcome drawbacks of conventional bilinear pooling: (1) The conventional bilinear pooling is designed for visual classification. Thus, it aggregates all the features globally, destroying the local data structure which is important for semantic segmentation. (2) The conventional bilinear pooling aggregates the outer products of the feature vectors by averaging, and hence loses representativeness when the real data distribution is complex. (3) The conventional bilinear pooling lifts the feature dimension from d to d^2, causing parameter proliferation in the

neural net and expensive computational cost.

In this work we extend the conventional bilinear pooling from several aspects and make it suitable for fine-grained action parsing. Specifically, we make the following contributions: (1) To enrich the representativeness, we decouple the first and second-order components from the bilinear form, and replace the averaging by convolution of a learnable filter. In this case, the proposed bilinear form is adaptive to the data and guided by the training objective. (2) To reduce the dimensionality *without* suffering from information loss or requiring extra computation, we propose lower-dimensional feature mappings than the explicit bilinear compositions. Such feature mapping is equivalent to the bilinear form, in the sense that the associated kernel function, and hence the *reproducing-kernel Hilbert space* (RKHS), is identical. (3) We perform extensive experiments to investigate our novel bilinear pooling methods, and show that the proposed method consistently improves or is on-par with the performance of the state-of-the-art methods on diverse datasets. To our knowledge, we are the first to employ bilinear pooling in a convolutional encoder-decoder architecture for fine-grained action parsing over time.

2 Related work

Fine-grained temporal action parsing. [10] proposes to learn object and material states, and partition actions by detecting the state transitions. [11] applies a statistical language model to capture action temporal dynamics. [12] proposes an Ego-ConvNet incorporating two streams for extracting spatial features and spatiotemporal features respectively from pre-defined video segments. The results are improved when combining Fisher vectors [13] from spatial and optical flow descriptors [14]. [15] proposes a multimodal bidirectional LSTM model to generate a label sequence of a video to incorporate forward and backward temporal dynamics. [16] proposes a conditional random field with skip connections in the temporal domain and starting-and-ending frame priors, which is learned via a structured support vector machine. [5] proposes a multi-modal deep neural net with the similar structure of the VGG net. After training and extracting frame-wise features, a temporal convolutional net and a semi-Markov conditional random field are applied to produce the final segmentation result. Based on the spatial features from [5], [6] proposes two kinds of temporal convolutional networks with the encoder-decoder architecture. The first net comprises layers of convolution and max pooling; the second net uses dilated temporal convolution and skipped connections to capture long-range temporal structures. Our work uses the temporal encoder-decoder architecture proposed by [6]. To capture second-order statistics, we replace the max pooling in [6] by our proposed bilinear pooling operations. We compare our method with others in Sec. 4. Although more complicated architectures, e.g. [17] [18], can also improve the performance, our work focuses on the pooling operation and hence investigating more advanced architectures is out of our scope.

Bilinear pooling. Bilinear pooling (or second-order pooling) is widely used in fine-grained visual classification [7–9, 19–33], visual questioning answering [34–36], feature fusion and disentangling [22, 23, 37–40], action recognition [39–44] and other tasks. In deep neural nets, bilinear pooling is mostly used only once before the classification layer, e.g. in [8, 9, 22, 23, 25, 26, 31, 33, 38, 39], or embedded within the classifier, e.g. in [27, 29].

There are three major research directions regarding bilinear pooling: (1) Dimension reduction while minimizing information loss. [8, 32, 41] use tensor sketch [45] to reduce the dimension of vectorized bilinear forms. The studies of [9, 23] use parametric dimension reduction approaches, which can be learned via back-propagation. The work in [35] [30] finds a low-rank approximation of the bilinear forms, so as to convert vector outer product into Hadamard multiplication for cheap computation. [19,24,29] utilize singular value decomposition (SVD), which can be used to select principle components and increase the performance at a higher computational cost. (2) Multiple bilinear pooling layers in deep neural nets. [40] factorizes bilinear composition into consecutive matrix multiplications along different dimensions. [30] uses the low-rank approximation as in [35], and aggregates features hierarchically. [46] fuses first and second-order information across layers to improve texture recognition. (3) Methods to capture richer feature statistics, so that more complex distributions can be represented. [47] proposes a higher-order pooling scheme to extract feature co-occurrences of visual words based on linearization of a higher-order polynomial kernel. [48] applies tensor sketch to generate a compact explicit feature map up to p-th order. Despite increasing the representativeness, more computational loads are caused. [44] linearizes a Gaussian kernel to derive a higher-order descriptor from the late fusion of CNN classifier scores for action recognition.

The novelties of our bilinear pooling method contribute to all the three research directions. First, we prove that our proposed bilinear forms correspond to feature mappings of some reproducing-kernel Hilbert spaces (RKHSs) endowed with polynomial kernels. We then find *exact* lower-dimensional alternative feature representations that retain the kernel evaluations in these RKHSs. As a result, the dimension can be reduced *without* information loss and additional computation. Second, our bilinear forms are used in multiple layers in the temporal convolutional encoder-decoder architecture, instead of being only used at the network top. Third, the first and second-order components

of the bilinear forms can be decoupled and each of them has different *learnable* weights. Despite staying in second-order, the learnable weights enable to create adaptive local statistics to the data, and hence can capture more complex statistics than the conventional bilinear pooling.

3 Method

3.1 Preliminaries

Temporal Convolutional Encoder-Decoder. The TCED net takes a temporal sequence of feature vectors and assigns an action label to each input feature vector. It comprises a stack of encoders and decoders, and a fully connected module to generate frame-wise action labels. Each encoder comprises a 1D temporal convolution layer with an activation function and a pooling layer to extract local statistics. After each encoder, the temporal resolution is halved. The decoder has a symmetric structure with the encoder, composed of a 1D temporal convolution layer and a upsampling layer to perform nearest-neighbor interpolation. After each decoder, the temporal resolution is doubled. The fully connected module incorporates a time-distributed fully connected layer to perform linear transformation at each time instant. Then each output is passed to a softmax function to fit the ground truth one-hot encoded action label. We refer to [6, Figure 1] for details.

Bilinear Pooling. Given a set of generic feature vectors with $\boldsymbol{x} \in \mathcal{X}$, the conventional bilinear pooling [7, 21–23] can be given by

$$\mathcal{B}(\mathcal{X}) = vec\left(\frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{x}\otimes\boldsymbol{x}\right), \tag{1}$$

where $\otimes$ denotes the vector outer product, $|\cdot|$ denotes the cardinality of the feature set and $vec(\cdot)$ denotes tensor vectorization. In this case, the bilinear composition gives a description of the feature set incorporating feature channel correlations.

3.2 Local Temporal Bilinear Composition

In contrast to many studies that perform global pooling for visual classification, we define the feature set in Eq. (1) as a local temporal neighborhood set to preserve the temporal structure. Specifically, given a temporal sequence of features $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_T\}$ with $\boldsymbol{x}_t \in \mathbb{R}^d$ for $t \in 1, 2, ..., T$, the local temporal bilinear composition which *couples* the first and second-order information is given by

$$\mathcal{B}_c(\boldsymbol{x}_t) = vec\left(\frac{1}{|\mathcal{N}(t)|}\cdot\sum_{\tau\in\mathcal{N}(t)} \boldsymbol{x}_\tau\otimes\boldsymbol{x}_\tau\right), \tag{2}$$

where $\mathcal{N}(t)$ denotes the local temporal neighborhood set centered at time t. As the averaging operation ignores the real distribution in $\mathcal{N}(t)$, we enrich the representativeness of bilinear pooling with the following two perspectives.

3.2.1 Decoupling First and Second-order Information

Inspired by a physical fact that the position and the velocity of an object in motion can indicate the dynamic state independently and complementarily, we consider to separate first and second-order components from the bilinear form to describe the action via separate attributes. Provided the feature time sequence $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_T\}$, the first-order component $\boldsymbol{\mu}$, the second-order component $\boldsymbol{\Sigma}$, and the decoupled bilinear form $\mathcal{B}_d(\cdot)$ are given by

$$\boldsymbol{\mu}_t = \frac{1}{|\mathcal{N}(t)|}\cdot\sum_{\tau\in\mathcal{N}(t)} \boldsymbol{x}_\tau, \tag{3}$$

$$\boldsymbol{\Sigma}_t = \frac{1}{|\mathcal{N}(t)|}\cdot\sum_{\tau\in\mathcal{N}(t)} (\boldsymbol{x}_\tau-\boldsymbol{\mu}_t)\otimes(\boldsymbol{x}_\tau-\boldsymbol{\mu}_t) \text{ and} \tag{4}$$

$$\mathcal{B}_d(\boldsymbol{x}_t) = \Big(\boldsymbol{\mu}_t^T, vec\big(\boldsymbol{\Sigma}_t\big)\Big)^T, \tag{5}$$

in which one can note $\mathcal{B}_d(\boldsymbol{x}_t) \in \mathbb{R}^{d(d+1)}$. Since the first-order component is equivalent to the mean and the second-order component is equivalent to the covariance, such decomposed bilinear form can precisely describe a Gaussian distribution.

3.2.2 Adapting local statistics to data

When the local statistics is more complex than Gaussian distribution, only using mean and covariance is not sufficient. Rather than applying higher-order statistics (e.g. [47, 48]), we consider statistics up to the second-order to retain a low computational load. Since the averaging operation in Eq. (2) and Eq. (3) can be regarded as convolution by a box filter, we generalize it to convolution by a learnable filter. Thus, the local statistics is adaptive to the data and the network objective. Specifically, for the coupled bilinear form the learnable version is given by

$$\mathcal{B}_c(\boldsymbol{x}_t) = vec\left(\sum_{\tau\in\mathcal{N}(t)} \omega_\tau\boldsymbol{x}_\tau\otimes\boldsymbol{x}_\tau\right), \tag{6}$$

where the filter weights $\{\omega_\tau\}$ are shared by all temporal neighbor sets, i.e. $\mathcal{N}(t)$ with $t = 1, 2, ..., T$.

For the decoupled bilinear form, the learnable version is given by

$$\boldsymbol{\mu}_t = \sum_{\tau \in \mathcal{N}(t)} p_\tau \boldsymbol{x}_\tau, \tag{7}$$

$$\boldsymbol{\Sigma}_t = \sum_{\tau \in \mathcal{N}(t)} q_\tau (\boldsymbol{x}_\tau - \boldsymbol{\mu}_t) \otimes (\boldsymbol{x}_\tau - \boldsymbol{\mu}_t) \text{ and} \tag{8}$$

$$\mathcal{B}_d(\boldsymbol{x}_t) = \left(\boldsymbol{\mu}_t^T, vec\big(\boldsymbol{\Sigma}_t\big)\right)^T, \tag{9}$$

where the filter weights $\{p_\tau\}$ and $\{q_\tau\}$ are shared by all temporal neighbor sets.

3.3 Normalization

Our bilinear forms are applied in several intermediate layers of the neural net. Due to the vector outer product, small values become smaller and large values become larger as the data flows from the net bottom to top, leading to diverging spectra in the bilinear forms and very sparse features before the final classification layer. Here we present three normalization methods that can constrain the bilinear form spectra or densify the features.

l_2 normalization. We can apply l_2 normalization after each bilinear pooling. Since the l_2 norm of a vectorized matrix is equivalent to its Frobenius norm and also equivalent to the Frobenius norm of the singular value matrix after SVD, the l_2 normalization on the vectorized bilinear form can constrain the matrix spectra between 0 and 1, and hence eliminates the diverging spectra problem.

Regularized power normalization. When using element-wise power normalization, as e.g. in [7, 19, 24], or matrix [19, 24] or higher-order tensor [19] spectral power normalization in intermediate layers of a neural net, gradients tend to explode during back-propagation when small or zero values are encountered. We propose a regularized version and use it as an activation function after each 1D convolution layer, so that features in the net are always densified. The formula is given by

$$\sigma(x) = \textbf{RPN}(x) = \text{sign}(x) \cdot \left(\sqrt{|x| + \theta^2} - \sqrt{\theta^2}\right), \tag{10}$$

where **RPN** stands for *regularized power normalization* and θ is a learnable parameter. As $\theta \to 0$, the **RPN** function converges to the standard power normalization. There exist many studies to make power normalization well-behaved, yet detailed discussion on such topic is out of our scope. One can see [49] for other smooth power normalization methods which are proposed for deep neural nets.

Normalized ReLU. [6] proposes a normalized ReLU activation function, which allows fast convergence and yields superior results to other activation functions. The formula is given by

$$\sigma(\boldsymbol{x}) = \textbf{NReLU}(\boldsymbol{x}) = \frac{ReLU(\boldsymbol{x})}{max(ReLU(\boldsymbol{x})) + \epsilon}, \tag{11}$$

where **NReLU** stands for normalized ReLU, ϵ is a small positive constant and the $max(\cdot)$ operation selects the maximal value in each feature vector. Since the Frobenius norm is bounded by the max norm of a matrix [50], **NReLU** is also able to constrain the matrix singular values and hence eliminates the diverging spectra issue. Nevertheless, it can lead to sparse features.

3.4 Low-dimensional Representation

Given an arbitrary feature vector sequence, the bilinear forms $\mathcal{B}_c$ and $\mathcal{B}_d$ can capture local temporal statistics which are adaptive to the data. However, the feature dimension is considerably increased. Specifically, given $\boldsymbol{x}_t \in \mathbb{R}^d$, we have $\mathcal{B}_c(\boldsymbol{x}_t) \in \mathbb{R}^{d^2}$ and $\mathcal{B}_d(\boldsymbol{x}_t) \in \mathbb{R}^{d(d+1)}$. To address such issue, we propose alternative lower-dimensional representations to the explicit bilinear forms defined in Eq. (6) and Eq. (9). Comparing to other dimension reduction methods introduced in Sec. 2, our method is *exact* which means it introduces *neither* information loss as in approximation methods nor additional computational costs as in SVD.

We first show that $\mathcal{B}_c(\cdot)$ and $\mathcal{B}_d(\cdot)$ are feature mappings associated with reproducing kernel Hilbert spaces (RKHSs) [51], for which the kernels are seconds-order homogeneous and inhomogeneous polynomials, respectively. Such property can be extended to arbitrary p-th order polynomials. One can see more details in [52, Chapter 3].

Proposition 1. *Given $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_T\}$, we have*

$$\langle \mathcal{B}_c(\boldsymbol{x}_i), \mathcal{B}_c(\boldsymbol{x}_j) \rangle_{\mathbb{R}^{d^2}} = \sum_{\tau \in \mathcal{N}(i)} \sum_{\tau' \in \mathcal{N}(j)} \omega_\tau \omega_{\tau'} \langle \boldsymbol{x}_\tau, \boldsymbol{x}_{\tau'} \rangle^2_{\mathbb{R}^d}, \tag{12}$$

and

$$\begin{aligned} &\langle \mathcal{B}_d(\boldsymbol{x}_i), \mathcal{B}_d(\boldsymbol{x}_j) \rangle_{\mathbb{R}^{d(d+1)}} = \langle \boldsymbol{\mu}_i, \boldsymbol{\mu}_j \rangle_{\mathbb{R}^d} \\ &+ \sum_{\tau \in \mathcal{N}(i)} \sum_{\tau' \in \mathcal{N}(j)} q_\tau q_{\tau'} \langle \boldsymbol{x}_\tau - \boldsymbol{\mu}_i, \boldsymbol{x}_{\tau'} - \boldsymbol{\mu}_j \rangle^2_{\mathbb{R}^{d^2}}, \end{aligned} \tag{13}$$

in which the notations are referred to the definitions in Eq. (6) and Eq. (9).

Proof. For the coupled bilinear composition, we have

$$\begin{aligned}
&\langle \mathcal{B}_c(\boldsymbol{x}_i), \mathcal{B}_c(\boldsymbol{x}_j) \rangle_{\mathbb{R}^{d^2}} \\
&= \left\langle \sum_{\tau \in \mathcal{N}(i)} vec\left(\omega_\tau \boldsymbol{x}_\tau \otimes \boldsymbol{x}_\tau\right), \sum_{\tau' \in \mathcal{N}(j)} vec\left(\omega_{\tau'} \boldsymbol{x}_{\tau'} \otimes \boldsymbol{x}_{\tau'}\right) \right\rangle \\
&= \sum_{\tau \in \mathcal{N}(i)} \sum_{\tau' \in \mathcal{N}(j)} \omega_\tau \omega_{\tau'} \left\langle vec\left(\boldsymbol{x}_\tau \otimes \boldsymbol{x}_\tau\right), vec\left(\boldsymbol{x}_{\tau'} \otimes \boldsymbol{x}_{\tau'}\right) \right\rangle \\
&= \sum_{\tau \in \mathcal{N}(i)} \sum_{\tau' \in \mathcal{N}(j)} \omega_\tau \omega_{\tau'} \left\langle \boldsymbol{x}_\tau, \boldsymbol{x}_{\tau'} \right\rangle^2_{\mathbb{R}^d}.
\end{aligned} \tag{14}$$

For the decoupled bilinear composition, we have

$$\begin{aligned}
\langle \mathcal{B}_d(\boldsymbol{x}_i), \mathcal{B}_d(\boldsymbol{x}_j) \rangle_{\mathbb{R}^{d(d+1)}} &= \langle \boldsymbol{\mu}_i, \boldsymbol{\mu}_j \rangle_{\mathbb{R}^d} \\
&+ \langle vec(\boldsymbol{\Sigma}_i), vec(\boldsymbol{\Sigma}_j) \rangle_{\mathbb{R}^{d^2}},
\end{aligned} \tag{15}$$

and hence can obtain Eq. (13) following the derivation in Eq. (14). □

One can see from Proposition 1 that the inner product defined w.r.t. $\mathcal{B}_c(\cdot)$ can be expressed in terms of the 2nd-degree homogeneous polynomial kernel $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle^2$. In general, the dimension of $\mathcal{B}_c(\cdot)$ increases *exponentially* with the degree of the polynomial kernel, making it less practical when used explicitly in a deep neural net. Motivated by the fact that for a specific kernel $k(\cdot, \cdot)$, the associated feature mapping $\phi : \boldsymbol{X} \to \mathcal{H}$ is not unique, we derive a feature mapping that corresponds to the same kernel as $\mathcal{B}_c(\cdot)$, but has lower dimension. The proposed method reduces the number of parameters to be learned without sacrificing the representativeness. In particular, we show that:

Proposition 2. *Let $\mathcal{B}_c(\boldsymbol{x}) \in \mathbb{R}^{d^2}$ be the bilinear composition and $\phi_c(\boldsymbol{x}) \in \mathbb{R}^{\frac{d(d+1)}{2}}$ a feature mapping defined by*

$$\phi_c(\boldsymbol{x}) = (\underbrace{x_1^2, ..., x_d^2}_{d \text{ terms}}, \underbrace{\sqrt{2}x_1x_2, \sqrt{2}x_1x_3, ..., \sqrt{2}x_{d-1}x_d}_{C(d,2) \text{ terms}})^T. \tag{16}$$

Then, it follows that for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$,

$$\langle \mathcal{B}_c(\boldsymbol{x}), \mathcal{B}_c(\boldsymbol{x}') \rangle_{\mathbb{R}^{d^2}} = \langle \phi_c(\boldsymbol{x}), \phi_c(\boldsymbol{x}') \rangle_{\mathbb{R}^{\frac{d(d+1)}{2}}}.$$

Equivalently, the second-order component defined in Eq. (9) has a lower-dimensional alternative, so that $\mathcal{B}_d(\boldsymbol{x}) \in \mathbb{R}^{d(d+1)}$ can be replaced by $\phi_d(\boldsymbol{x}) \in \mathbb{R}^{\frac{d(d+3)}{2}}$.

Due to the commutative property of tensor product, the above proposition can be proved by expanding the polynomials in Eq. (14) and combining equivalent terms.

Proposition 2 shows that $\mathcal{B}_c(\cdot)$ and $\phi_c(\cdot)$ are equivalent in the sense that the corresponding kernel is the same. The advantage of using $\phi_c(\cdot)$ instead of $\mathcal{B}_c(\cdot)$ is that it has much lower dimension. For example, if each feature vector in the input sequence is 128-dimensional, $\mathcal{B}_c$ is 16384-dimensional and $\mathcal{B}_d$ is 16512-dimensional. On the other hand, the alternative feature representations ϕ_c is 8256-dimensional and ϕ_d is 8384-dimensional, approximately halving the dimensionality without losing information and without introducing extra computation.

4 Experiment

4.1 Datasets

In our experiments, the input feature time sequence to the TCED net is extracted from RGB videos using a pre-trained VGG-like network [5], and is downsampled to achieve the same temporal resolution as [6] for fair comparison.

50 Salads [53]. This multi-modal dataset collects 50 recordings from 25 people preparing 2 mixed salads, and each recording lasts 5-10 minutes. The RGB video has spatial resolution of 640x480 pixels and frame rate of 30 fps. The annotation is performed at two levels: (1) the *eval-level* incorporating 9 actions such as "cut", "peel" and "add dressing", and (2) the *mid-level* incorporating 17 fine-grained actions, derived from the high-level actions. Therefore, we obtain two sets from **50 Salads**, namely **50 Salads-eval** and **50 Salads-mid**. The recordings are equally split into 5 folds for cross-validation.

Georgia Tech Egocentric Activity Datasets (GTEA) [54] [1]. This dataset contains 7 daily living activities of 4 subjects. The videos are captured from the egocentric view at 15 fps with the resolution of 1280x720 pixels and there are 31,222 frames in the dataset. We follow the settings in [5] [6]: For each video, frame-wise labels from 11 action classes are annotated. The evaluation is based on the leave-one-subject-out scheme, namely performing cross-validation on 4-fold splits.

JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) [55] [2]. In our study we only use the videos of "suturing" since it has more trials than other tasks. The "suturing" task comprises 10 actions like "tie a knot", "insert needle into skin" and so forth. Each video is approx. 2 minutes and contains 15 to 37 actions, which have considerably different occurrence orders from different surgeons. Similar to GTEA, in our experiments we perform evaluations in the leave-one-surgeon-out scheme.

4.2 Evaluation Metrics

Frame-wise accuracy. The frame-wise accuracy is defined as the correctly classified frames divided by the number of all frames. Intuitively, such measure evaluates the accuracy from the frame-wise classification perspective. However, it

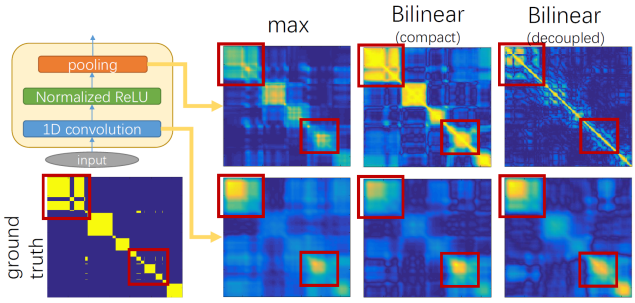

Figure 1. We use the features from the first encoder of TCED to show frame similarities of "rgb-01-1.avi" in **50 Salads-mid** [53]. The similarity of two frame features $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as $|\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle|$. The similarity of two (one-hot) frame labels, regarded as ground truth, is computed in the same way. The bilinear pooling outputs are power-and-l2 normalized. Entries in similarity matrices range between 0 (blue) and 1 (yellow). Red rectangles contain some fine-grained actions. One can see that bilinear pooling is better at recognizing fine-grained actions, but can decompose coarse-grained actions.

ignores the temporal regularity and the action occurrence order in the label sequence.

Edit score [5]. The edit score evaluates the temporal order of action occurrence, ignores the action temporal durations and only considers segment insertions, deletions and substitutions. Thus, such metric is useful for scenarios, where the action order is essential, e.g. cooking, manufacturing, surgery and so forth. However, the edit score can be strongly penalized by tiny predicted segments, and hence highly degraded by over-segmentation results.

F1 score [6]. The F1 score is for evaluation in terms of action detection, where the true positives are defined by segments whose action label is same to the ground truth and the *intersection-over-union* of the overlap with the ground truth is greater than 0.1. Thus, it is invariant to small temporal shifts between detection and the ground truth. However, the F1-score is penalized by over-segmentation as well, since lots of tiny segments can result in a low precision rate.

4.3 Analysis of the Bilinear Forms

We use the **50 Salads-mid** dataset to perform model analysis, because it has more fine-grained action types and longer video recordings than the other mentioned datasets.

The benefits of second-order information. Fig. 1 illustrates the comparisons between pooling methods, where the compact bilinear pooling [8] output has the **same** dimension as the max pooling: (1) The first row in Fig. 1 clearly shows that bilinear pooling can capture fine-grained actions better than max pooling, whose output features tend to merge fine-grained actions into coarse-grained ones. Our proposed decoupled bilinear pooling with full second-order information performs better at recognizing fine-grained actions and suppressing off-diagonal elements. However, it can break a coarse-grained action into several segments. The compact bilinear pooling outperforms max pooling on the diagonal elements, which can clearly show that the advantage of bilinear pooling is due to the second-order information rather than higher dimensionality. However, the large off-diagonal values indicate the drawback of the dimension reduction method with approximation. (2) The second row in Fig. 1 illustrates that bilinear pooling improves the convolution layer via backpropagation. With max pooling, many off-diagonal elements are similar to the diagonal elements, which differ from the ground truth pattern considerably. However, with bilinear pooling, the matrix patterns are more similar to the ground truth.

Furthermore, we conduct a quantitative comparison on the *first split* of **50 Salads-mid**. In the format of *accuracy/edit-score/F1-score*, max pooling yields 71.03/71.8/73.09, compact bilinear pooling yields 75.41/73.75/78.96, coupled bilinear pooling yields 76.56/75.32/79.84 and decoupled bilinear pooling yields 75.11/71.06/75.79. One can see that bilinear pooling consistently outperforms max pooling. The comparison between max pooling and compact bilinear pooling also indicates the importance of the second-order information.

Comparison of different bilinear forms. Here we analyze the influence of the learnable weights in the proposed bilinear forms $\mathcal{B}_c$ and $\mathcal{B}_d$. We denote the corresponding non-learnable bilinear forms in Eq. (2) and Eq. (3) as $\mathcal{B}_c^o$ and $\mathcal{B}_d^o$, respectively. As shown in the top row of Fig. 2, both for the coupled and decoupled bilinear forms, the one with learnable weights consistently outperforms the non-learnable counterpart, in terms of the evaluation metrics and the robustness to the neighborhood size $|\mathcal{N}|$. This outcome is more obvious when the neighbor size is larger. This result can indicate that the learnable weights, i.e. $\{\omega_\tau\}$, $\{p_\tau\}$ and $\{q_\tau\}$ in equations (6) and (9), enable the derived bilinear forms to capture more complex local temporal statistics, comparing to the standard average aggregation. Thus, in the following experiments, we only use the learnable bilinear forms. Furthermore, the decoupled bilinear form outperforms the coupled version on all the three metrics. Specifically, the decoupled bilinear form achieves *66.3/64.63/70.74* in the format of *accuracy/edit score/F1 score*, while the best performance of the coupled bilinear form is *64.73/62.15/68.89* and the baseline model (TCN$_{max}$ [6]) achieves *64.7/59.8/68.0*.

In the bottom row of Fig. 2, we show the performance of the first-order component and the second-order component of the decoupled bilinear form. One can observe that the results derived using individual components are inferior to the results using combined bilinear forms. This fits our conjecture that first and second-order components tend to describe independent and complementary patterns in data.

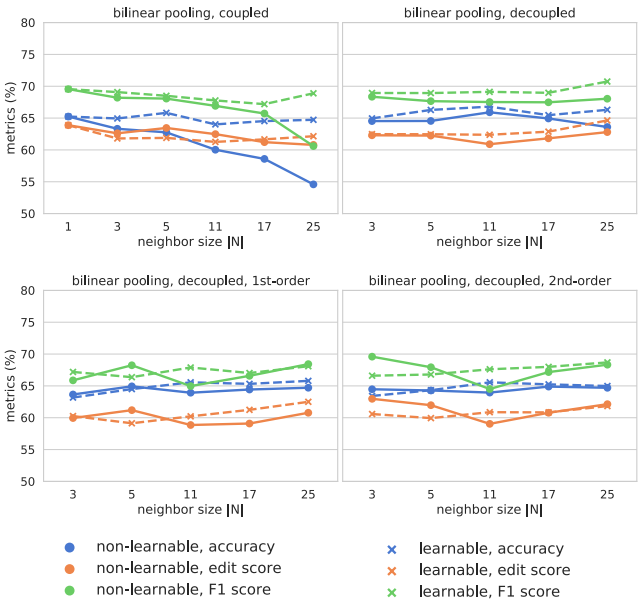

Figure 2. The performances w.r.t. the neighbor size $|\mathcal{N}|$ and the learnability of the weights. From top to bottom: (1) The performances of the coupled bilinear form $\mathcal{B}_c$ and the decoupled bilinear form $\mathcal{B}_d$. (2) The performances of each ingredient in the decoupled bilinear form $\mathcal{B}_d$, in which the first-order component and the second-order component are demonstrated in Eq. 9.

Normalization and activation. Here we investigate the influence of normalization and compare different activation functions. In each individual experiment the neighborhood size of both bilinear forms are identical. First, different normalization methods are compared in Tab. 1. One can see that l_2 normalization and l_1 normalization perform almost equally, while the normalized ReLU activation function consistently outperforms others. This result indicate that the max-normalization in intermediate layers is more suitable than others to constrain the bilinear form spectrum. Second, we show the influence of the activation functions in Tab. 2. The bilinear forms are l_2 normalized, except for the case of **NReLU**. In our experiment, training with other activation functions without l_2 normalization hardly converges, indicating the importance of constraining the spectrum of the bilinear form. Tab. 2 indicates that the **NReLU** function consistently yields superior results, suggesting that our task benefits from the sparse features.

	$\mathcal{B}_c$	$\mathcal{B}_d$
NReLU	**65.82/61.89/68.5**	**66.28/62.46/68.93**
NReLU+l_2	64.92/60.01/67.33	66.09/60.02/67.38
NReLU+l_1	64.22/60.04/65.86	66.48/63.04/68.71
ReLU+l_2	64.87/59.88/66.31	64.7/61.45/68.04
ReLU+l_1	63.05/58.29/65.62	59.76/58.68/64.39

Table 1. Comparison of different normalization methods, in which the performances are presented in terms of *accuracy/edit score/F1 score* and the best ones for each model are highlighted in boldface.

4.4 Low-dimensional Representation

One of our key contributions is to derive lower-dimensional alternatives to the explicit bilinear compositions. Comparing other dimension reduction methods, our method does not suffer from any information loss nor do we have extra computation. We compare different low-dimensional representations in the lower parts of Table 3, 4, 5 and 6. The *tensor sketch* technique [8] reduces each feature outer product from d^2 to $\frac{d(d+1)}{2}$ for fair comparison. In addition, the *LearnableProjection* [23] is implemented by a temporal convolution layer with the kernel size of 1, and the reduced dimensions are equal to ϕ_c and ϕ_d respectively for fair comparison. Note that, in our trials, other dimension reduction methods (especially the ones employing SVD) used in our local temporal pooling cause very high computational cost, and hence are not compared. For each listed method we tested different neighborhood sizes of 5, 11 and 25, and present the best performance. Our results show that the proposed low-dimensional representations consistently outperform other dimension reduction methods. In particular, on the **50 Salads-mid** dataset, ϕ_d considerablely outperforms the *LearnableProjection* counterpart, in which the accuracy is improved by 5.6%, the edit score is improved by 6.2% and the F1 score is improved by 6.1%.

4.5 Comparison with State-of-the-art

Table 3, 4, 5 and 6 show the performances of different methods on the datasets **50 Salads-mid**, **50 Salads-eval**, **GTEA** and **JIGSAWS**, respectively, in which TCED$_X$ denotes the temporal convolutional encoder-decoder with the pooling method X. For each method with local temporal pooling, we perform grid search on the neighborhood sets of 5, 11 and 25, and present the best one. From the tables, we can see that our proposed method can be generalized well across different datasets and produces superior or comparable performances than other methods. In **50 Salads-mid**, the dataset with more fine-grained action types and longer videos than other datasets, the decoupled bilinear form, as well as its lower-dimensional representation outperform other methods for all the evaluation metrics. In **50 Salads-eval**, the performance of our methods are comparable with others while with lower edit scores, probably because actions in this dataset is not sufficiently fine-grained but our bilinear pooling produces more segments than others. Furthermore, more training epochs can increase the accuracy yet decrease the edit score and the F1 score for our bilinear pooling models, in contrast to the max pooling baseline model. For example, after 300 epochs, TCED$_{\mathcal{B}_d}$ yields 74.7/59.2/66.7 and TCED$_{max}$ yields 63.6/71.9/75.2 for the **GTEA** dataset.

	ReLU [56]	**leaky ReLU** [57]	**swish** [58]	**NReLU (Eq. 11)** [6]	**RPN (Eq. 10)**	**linear**
max	61.13/53.13/59.78	54.97/48.51/55.58	56.51/47.06/52.39	**63.55/60.37/64.88**	62.65/54.89/63.05	12.59/11.51/8.63
$\mathcal{B}_c$	65.5/61.14/68.4	66.4/**61.72/69.08**	**66.77**/59.16/67.49	64.01/61.26/67.77	64.05/56.48/64.77	63.47/48.34/55.87
$\mathcal{B}_d$	64.7/61.45/68.04	65.56/53.55/61.45	62.51/49.26/56.64	**66.8/62.38/69.12**	63.18/50.41/58.39	65.51/48.31/56.96

Table 2. The performances with different pooling methods and activation functions are presented in the format of *accuracy/edit score/F1 score*, in which for each model the best results are highlighted in boldface.

Method	**Result**
Spatial CNN [5]	54.9/24.8/32.3
Spatiotemporal CNN [5]	59.4/45.9/55.9
IDT+LM [11]	48.7/45.8/44.4
Dilated TCN [6]	59.3/43.1/52.2
Bidirectional LSTM [6]	55.7/55.6/62.6
TCED$_{max}$ [6]	64.7/59.8/68.0
TCED$_{\mathcal{B}_c}$	65.8/61.9/68.5
TCED$_{\mathcal{B}_d}$	**66.3**/62.5/68.9
TCED$_{TensorSketch}$ [8]	63.4/62.6/68.5
TCED$_{\mathcal{B}_c,LearnableProjection}$	61.8/58.2/64.4
TCED$_{\mathcal{B}_d,LearnableProjection}$	60.1/56.6/62.9
TCED$_{\phi_c}$	64.7/61.3/66.8
TCED$_{\phi_d}$	65.7/**62.8/69.0**

Table 3. The comparison in **50 Salads-mid**, where the results are shown in the format of *accuracy/edit score/F1 score*. The upper part shows the comparison with other action parsing methods and the lower part shows the comparison of different dimension reduction methods. The best results are highlighted in boldface.

Method	**Result**
Spatial CNN [5]	68.0/25.5/35.0
Spatiotemporal CNN [5]	71.3/52.8/61.7
Dilated TCN [6]	71.1/46.9/55.8
Bidirectional LSTM [6]	70.9/67.7/72.2
TCED$_{max}$ [6]	73.4/**72.2/76.5**
TCED$_{\mathcal{B}_c}$	74.2/71.2/75.5
TCED$_{\mathcal{B}_d}$	**75.9**/71.3/76.2
TCED$_{TensorSketch}$ [8]	71.9/70.9/75.1
TCED$_{\mathcal{B}_c,LearnableProjection}$	72.0/68.8/73.4
TCED$_{\mathcal{B}_d,LearnableProjection}$	71.3/68.9/72.6
TCED$_{\phi_c}$	74.0/71.0/**76.5**
TCED$_{\phi_d}$	75.6/70.4/76.0

Table 4. The comparison in **50 Salads-eval**.

Method	**Result**
EgoNet+TDD [12]	64.4/-/-
Spatial CNN [5]	54.8/28.7/38.3
Spatiotemporal CNN [5]	57.6/49.1/56.7
Spatiotemporal CNN+Seg [5]	52.6/53.0/57.7
Dilated TCN [6]	58.0/40.7/51.3
Bidirectional LSTM [6]	56.2/41.3/50.2
TCED$_{max}$ [6]	63.5/71.9/75.2
TCED$_{\mathcal{B}_c}$	63.6/71.7/76.4
TCED$_{\mathcal{B}_d}$	63.4/70.9/**76.8**
TCED$_{TensorSketch}$ [8]	59.8/71.2/75.2
TCED$_{\mathcal{B}_c,LearnableProjection}$	58.4/68.2/71.9
TCED$_{\mathcal{B}_d,LearnableProjection}$	58.8/70.5/74.9
TCED$_{\phi_c}$	**64.5**/71.8/75.0
TCED$_{\phi_d}$	64.4/**73.9**/76.3

Table 5. The comparison in **GTEA**, in which the symbol "-" denotes that the score is not available.

Method	**Result**
Spatial CNN [5]	74.1/37.7/51.6
Spatiotemporal CNN [5]	77.9/67.1/77.7
Spatiotemporal CNN+Seg [5]	74.4/73.7/82.2
Dilated TCN [6]	78.0/56.8/69.7
Bidirectional LSTM [6]	74.4/73.7/82.2
TCED$_{max}$ [6]	81.2/85.6/90.3
TCED$_{\mathcal{B}_c}$	**82.6**/85.6/90.4
TCED$_{\mathcal{B}_d}$	82.2/**87.7/91.4**
TCED$_{TensorSketch}$ [8]	80.8/85.4/90.1
TCED$_{\mathcal{B}_c,LearnableProjection}$	79.7/82.8/88.1
TCED$_{\mathcal{B}_d,LearnableProjection}$	81.6/83.0/89.0
TCED$_{\phi_c}$	81.8/85.1/90.0
TCED$_{\phi_d}$	81.7/85.1/90.5

Table 6. The comparison in **JIGSAWS**.

5 Conclusion

To our knowledge, we are the first to use bilinear pooling to a temporal convolutional encoder-decoder for action parsing. To enrich representativeness, we decouple the first and the second-order information from the conventional bilinear form and modify the averaging operation to convolution with a learnable filter. To reduce dimensionality, we introduce lower-dimensional representations of the bilinear forms with neither information loss nor extra computation. We conduct several detailed experiments to analyze the bilinear forms, and show superior performances to state-of-the-art pooling methods for action parsing. A future work is to investigate higher-order pooling with information lossless dimension reduction approaches.

Acknowledgements. Y. Z. and H. N. acknowledge funding by the BMBF project SenseEmotion. S. T. acknowledges funding by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Projektnummer 276693517 SFB 1233. We faithfully acknowledge Dr. Colin Lea (Facebook) to provide frame-wise features of the datasets.

References

[1] Yin Li, Zhefan Ye, and James M Rehg. Delving into egocentric actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 287–295, 2015. 1, 5

[2] Narges Ahmidi, Lingling Tao, Shahin Sefati, Yixin Gao, Colin Lea, Benjamin Bejar Haro, Luca Zappella, Sanjeev Khudanpur, René Vidal, and Gregory D Hager. A dataset and benchmarks for segmentation and recognition of gestures in robotic surgery. *IEEE Transactions on Biomedical Engineering*, 64(9):2025–2041, 2017. 1, 5

[3] Yan Zhang, He Sun, Siyu Tang, and Heiko Neumann. Temporal human action segmentation via dynamic clustering. *arXiv preprint arXiv:1803.05790*, 2018. 1

[4] Alexander Mathis, Pranav Mamidanna, Kevin M Cury, Taiga Abe, Venkatesh N Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, pages 1281 – 1289, September 2018. 1

[5] Colin Lea, Austin Reiter, René Vidal, and Gregory D Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. In *European Conference on Computer Vision*, pages 36–52. Springer, 2016. 1, 2, 5, 6, 8

[6] 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)*, pages 1003–1012, July 2017. 1, 2, 3, 4, 5, 6, 8

[7] Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *European Conference on Computer Vision*, pages 430–443. Springer, 2012. 1, 2, 3, 4

[8] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 317–326, 2016. 1, 2, 6, 7, 8

[9] Kaicheng Yu and Mathieu Salzmann. Statistically-motivated second-order pooling. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1, 2

[10] Alireza Fathi and James M Rehg. Modeling actions through state changes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2586, 2013. 2

[11] Alexander Richard and Juergen Gall. Temporal action detection using a statistical language model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3131–3140, 2016. 2, 8

[12] Suriya Singh, Chetan Arora, and CV Jawahar. First person action recognition using deep learned descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2620–2628, 2016. 2, 8

[13] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *2013 IEEE International Conference on Computer Vision*, pages 1817–1824, Dec 2013. 2

[14] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4305–4314, 2015. 2

[15] Bharat Singh, Tim K Marks, Michael Jones, Oncel Tuzel, and Ming Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1961–1970. IEEE, 2016. 2

[16] Colin Lea, René Vidal, and Gregory D Hager. Learning convolutional action primitives for fine-grained action recognition. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1642–1649. IEEE, 2016. 2

[17] Peng Lei and Sinisa Todorovic. Temporal deformable residual networks for action segmentation in videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[18] Khoi-Nguyen C Mac, Dhiraj Joshi, Raymond A Yeh, Jinjun Xiong, Rogerio R Feris, and Minh N Do. Locally-consistent deformable convolution networks for fine-grained action detection. *arXiv preprint arXiv:1811.08815*, 2018. 2

[19] P. Koniusz, F. Yan, P. Gosselin, and K. Mikolajczyk. Higher-order Occurrence Pooling on Mid- and Low-level Features: Visual Concept Detection. *Technical Report*, 2013. 2, 4

[20] Mohammad Moghimi, Serge J Belongie, Mohammad J Saberian, Jian Yang, Nuno Vasconcelos, and Li-Jia Li. Boosted convolutional neural networks. In *BMVC*, 2016. 2

[21] 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*, pages 2070–2078, 2017. 2, 3

[22] 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*, pages 1449–1457, 2015. 2, 3

[23] 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, 2018. 2, 3, 7

[24] Tsung-Yu Lin and Subhransu Maji. Improved bilinear pooling with cnns. *arXiv preprint arXiv:1707.06772*, 2017. 2, 4

[25] Tsung-Yu Lin, Subhransu Maji, and Piotr Koniusz. Second-order democratic aggregation. In *ECCV*, pages 639–656, 2018. 2

[26] Yumin Suh, Jingdong Wang, Siyu Tang, Tao Mei, and Kyoung Mu Lee. Part-aligned bilinear representations for person re-identification. In *European Conference on Computer Vision (ECCV)*, September 2018. 2

[27] Shu Kong and Charless Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7025–7034. IEEE, 2017. 2

[28] Peihua Li, Jiangtao Xie, Qilong Wang, and Zilin Gao. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. *arXiv preprint arXiv:1712.01034*, 2017. 2

[29] 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)*, pages 355–370, 2018. 2

[30] Chaojian Yu, Xinyi Zhao, Qi Zheng, Peng Zhang, and Xinge You. Hierarchical bilinear pooling for fine-grained visual recognition. In *European Conference on Computer Vision*, pages 595–610. Springer, 2018. 2

[31] Qilong Wang, Peihua Li, and Lei Zhang. G2denet: Global gaussian distribution embedding network and its application to visual recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3, 2017. 2

[32] 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*, pages 3175–3183, 2018. 2

[33] Marcel Simon, Yang Gao, Trevor Darrell, Joachim Denzler, and Erik Rodner. Generalized orderless pooling performs implicit salient matching. In *International Conference on Computer Vision (ICCV)*, 2017. 2

[34] 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 preprint arXiv:1606.01847*, 2016. 2

[35] 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. 2

[36] Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. Multimodal factorized bilinear pooling with co-attention learning for visual question answering. In *Proc. IEEE Int. Conf. Comp. Vis*, volume 3, 2017. 2

[37] Joshua B Tenenbaum and William T Freeman. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283, 2000. 2

[38] 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*, volume 1, 2017. 2

[39] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1933–1941, 2016. 2

[40] Jian-Fang Hu, Wei-Shi Zheng, Jiahui Pan, Jianhuang Lai, and Jianguo Zhang. Deep bilinear learning for rgb-d action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 335–351, 2018. 2

[41] Kaiyu Yue, Ming Sun, Yuchen Yuan, Feng Zhou, Errui Ding, and Fuxin Xu. Compact generalized non-local network. *arXiv preprint arXiv:1810.13125*, 2018. 2

[42] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *arXiv preprint arXiv:1711.07971*, 10, 2017. 2

[43] Rohit Girdhar and Deva Ramanan. Attentional pooling for action recognition. In *Advances in Neural Information Processing Systems*, pages 34–45, 2017. 2

[44] 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)*, pages 130–138. IEEE, 2017. 2

[45] 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*, pages 239–247. ACM, 2013. 2

[46] Xiyang Dai, Joe Yue-Hei Ng, and Larry S Davis. Fason: First and second order information fusion network for texture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7352–7360, 2017. 2

[47] 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. 2, 3

[48] Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge J Belongie. Kernel pooling for convolutional neural networks. In *CVPR*, volume 1, page 7, 2017. 2, 3

[49] Piotr Koniusz, Hongguang Zhang, and Fatih Porikli. A deeper look at power normalizations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5774–5783, 2018. 4

[50] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012. 4

[51] Bernhard Schölkopf and Alexander Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2002. 4

[52] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends in Machine Learning*, 10(1-2):1–141, 2017. 4

[53] 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*, pages 729–738. ACM, 2013. 5, 6

[54] Alireza Fathi, Xiaofeng Ren, and James M Rehg. Learning to recognize objects in egocentric activities. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference On*, pages 3281–3288. IEEE, 2011. 5

[55] Yixin Gao, S Swaroop Vedula, Carol E Reiley, Narges Ahmidi, Balakrishnan Varadarajan, Henry C Lin, Lingling Tao, Luca Zappella, Benjamın Béjar, David D Yuh, et al. Jhu-isi gesture and skill assessment working set (jigsaws): A surgical activity dataset for human motion modeling. In *MICCAI Workshop: M2CAI*, volume 3, page 3, 2014. 5

[56] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 8

[57] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013. 8

[58] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017. 8