Local Temporal Bilinear Pooling for Fine-Grained Action Parsing

Yan Zhang\textsuperscript{1,2}, Siyu Tang\textsuperscript{2,3}, Krikamol Muandet\textsuperscript{2}, Christian Jarvers\textsuperscript{1}, and Heiko Neumann\textsuperscript{1}

\textsuperscript{1}Institute of Neural Information Processing, Ulm University, Ulm, Germany
\textsuperscript{2}Max Planck Institute for Intelligent Systems, Tübingen, Germany
\textsuperscript{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 a label action 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 generate frame-wise action labels. 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 \( \mathbf{x} \in \mathcal{X} \), the conventional bilinear pooling [7, 21–23] can be given by

\[
\mathcal{B}(\mathcal{X}) = vec\left( \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} \mathbf{x} \otimes \mathbf{x} \right),
\]

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 \( \mathbf{X} = \{ \mathbf{x}_1, ..., \mathbf{x}_T \} \) with \( \mathbf{x}_t \in \mathbb{R}^d \) for \( t = 1, 2, ..., T \), the local temporal bilinear composition which couples the first and second-order information is given by

\[
\mathcal{B}_c(\mathbf{x}_t) = vec\left( \frac{1}{|\mathcal{N}(t)|} \sum_{\tau \in \mathcal{N}(t)} \mathbf{x}_\tau \otimes \mathbf{x}_\tau \right),
\]

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 \( \{ \mathbf{x}_1, ..., \mathbf{x}_T \} \), the first-order component \( \mathbf{\mu} \), the second-order component \( \mathbf{\Sigma} \), and the decoupled bilinear form \( \mathcal{B}_d(\cdot) \) are given by

\[
\mathbf{\mu}_t = \frac{1}{|\mathcal{N}(t)|} \sum_{\tau \in \mathcal{N}(t)} \mathbf{x}_\tau, \quad (3)
\]

\[
\mathbf{\Sigma}_t = \frac{1}{|\mathcal{N}(t)|} \sum_{\tau \in \mathcal{N}(t)} (\mathbf{x}_\tau - \mathbf{\mu}_t) \otimes (\mathbf{x}_\tau - \mathbf{\mu}_t) \quad (4)
\]

\[
\mathcal{B}_d(\mathbf{x}_t) = \left( \mu_t^T, vec(\mathbf{\Sigma}_t) \right)^T, \quad (5)
\]

in which one can note \( \mathcal{B}_d(\mathbf{x}_t) \in \mathbb{R}^{d(d+1)} \). 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

\[
\mathcal{B}_c(\mathbf{x}_t) = vec\left( \sum_{\tau \in \mathcal{N}(t)} \omega_\tau \mathbf{x}_\tau \otimes \mathbf{x}_\tau \right),
\]

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
\[
\mu_t = \sum_{\tau \in \mathcal{N}(t)} p_\tau x_\tau,
\]
\[
\Sigma_t = \sum_{\tau \in \mathcal{N}(t)} q_\tau (x_\tau - \mu_t) \otimes (x_\tau - \mu_t)
\] and
\[
\mathcal{B}_d(x_i) = \left(\mu^T_t, \text{vec}(\Sigma_t)\right)^T,
\]
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 \(l_2\) norm 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
\[
\sigma(x) = \text{RPN}(x) = \text{sign}(x) \cdot \left(\sqrt{|x^2|} + \theta^2 - \sqrt{\theta^2}\right), \tag{10}
\]
where RPN stands for regularized power normalization and \(\theta\) 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
\[
\sigma(x) = \text{NReLU}(x) = \frac{\text{ReLU}(x)}{\max(\text{ReLU}(x)) + \epsilon}, \tag{11}
\]
where \(\text{NReLU}\) stands for normalized ReLU, \(\epsilon\) 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], \(\text{NReLU}\) 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 \(x_i \in \mathbb{R}^d\), we have \(\mathcal{B}_c(x_i) \in \mathbb{R}^{d^2}\) and \(\mathcal{B}_d(x_i) \in \mathbb{R}^{d(d+1)}\). 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 second-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 \(\{x_1, ..., x_T\}\), we have
\[
\langle \mathcal{B}_c(x_i), \mathcal{B}_c(x_j) \rangle_{\mathbb{R}^{d^2}} = \sum_{\tau \in \mathcal{N}(i)} \sum_{\tau' \in \mathcal{N}(j)} \omega_{\tau \tau'} \langle x_\tau, x_{\tau'} \rangle_{\mathbb{R}^d}^2, \tag{12}
\]
and
\[
\langle \mathcal{B}_d(x_i), \mathcal{B}_d(x_j) \rangle_{\mathbb{R}^{d(d+1)}} = \langle \mu_i, \mu_j \rangle_{\mathbb{R}^d} + \sum_{\tau \in \mathcal{N}(i)} \sum_{\tau' \in \mathcal{N}(j)} q_{\tau \tau'} \langle x_\tau - \mu_i, x_{\tau'} - \mu_j \rangle_{\mathbb{R}^d}^2, \tag{13}
\]
in which the notations are referred to the definitions in Eq. (6) and Eq. (9).
that the inner product defined w.r.t. \( B_c(x, x') \) can be expressed in terms of the polynomial kernel \( k(x, x') = \langle x, x' \rangle^2 \). In general, the dimension of \( B_c(\cdot) \) 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 \( \phi : X \rightarrow \mathcal{H} \) is not unique, we derive a feature mapping that corresponds to the same kernel as \( B_c(\cdot) \), 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 \( B_c(x) \in \mathbb{R}^{d^2} \) be the bilinear composition and \( \phi_c(x) \in \mathbb{R}^{\frac{d(d+1)}{2}} \) be a feature mapping defined by

\[
\phi_c(x) = \begin{pmatrix} x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \ldots, \sqrt{2}x_{d-1}x_d \end{pmatrix}^T.
\]

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

\[
\langle B_c(x), B_c(x') \rangle_{\mathbb{R}^{d^2}} = \langle \phi_c(x), \phi_c(x') \rangle_{\mathbb{R}^{\frac{d(d+1)}{2}}}.
\]

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

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 \( B_c(\cdot) \) 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 \( B_c(\cdot) \) is that it has much lower dimension. For example, if each feature vector in the input sequence is 128-dimensional, \( B_c \) is 16384-dimensional and \( B_d \) is 16512-dimensional. On the other hand, the alternative feature representations \( \phi_c \) is 8256-dimensional and \( \phi_d \) 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
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 $B_c$ and $B_d$. We denote the corresponding non-learnable bilinear forms in Eq. (2) and Eq. (3) as $B^c$ and $B^d$, 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 $|N|$. This outcome is more obvious when the neighbor size is larger. This result can indicate that the learnable weights, i.e. $\{\omega_T\}$, $\{p_T\}$ and $\{q_T\}$ 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.

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. The similarity of two frame features $x_i$ and $x_j$ is defined as $\langle x_i, x_j \rangle$. 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.

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 $x_i$ and $x_j$ is defined as $\langle x_i, x_j \rangle$. 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.

|⟨x_i, x_j⟩| of the overlap with the ground truth dimension.

|N| . This outcome is more obvious when the neighbor size is larger. This result can indicate that the learnable weights, i.e. $\{\omega_T\}$, $\{p_T\}$ and $\{q_T\}$ 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.
4.4 Low-dimensional Representation

One of our key contributions is to derive low-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 $\phi_c$ and $\phi_d$, 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, $\phi_d$ considerably 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 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 is 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$_{\phi_c}$ yields 74.7/59.2/66.7 and TCED$_{\phi_d}$ yields 63.6/71.9/75.2 for the GTEA dataset.
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.

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.

### Table 2

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial CNN [5]</td>
<td>54.9/24.8/32.3</td>
</tr>
<tr>
<td>Spatiotemporal CNN [5]</td>
<td>59.4/45.9/55.9</td>
</tr>
<tr>
<td>IDT+LM [11]</td>
<td>48.7/45.8/44.4</td>
</tr>
<tr>
<td>Dilated TCN [6]</td>
<td>59.3/43.1/52.2</td>
</tr>
<tr>
<td>Bidirectional LSTM [6]</td>
<td>55.7/55.6/62.6</td>
</tr>
<tr>
<td>TCEDmax [6]</td>
<td>64.7/59.8/68.0</td>
</tr>
<tr>
<td>TCEDa</td>
<td>65.8/61.9/68.5</td>
</tr>
<tr>
<td>TCEDb</td>
<td>66.3/62.5/68.9</td>
</tr>
<tr>
<td>TCEDTensorSketch [8]</td>
<td>63.4/62.6/68.5</td>
</tr>
<tr>
<td>TCEDa,LearnableProjection</td>
<td>61.8/58.2/64.4</td>
</tr>
<tr>
<td>TCEDb,LearnableProjection</td>
<td>60.1/56.6/62.9</td>
</tr>
<tr>
<td>TCEDa</td>
<td>64.7/61.3/66.8</td>
</tr>
<tr>
<td>TCEDb</td>
<td>65.7/62.8/69.0</td>
</tr>
</tbody>
</table>

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.

### Table 4

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial CNN [5]</td>
<td>68.0/25.5/35.0</td>
</tr>
<tr>
<td>Spatiotemporal CNN [5]</td>
<td>71.3/52.8/61.7</td>
</tr>
<tr>
<td>Dilated TCN [6]</td>
<td>71.1/46.9/55.8</td>
</tr>
<tr>
<td>Bidirectional LSTM [6]</td>
<td>70.9/67.7/72.2</td>
</tr>
<tr>
<td>TCEDmax [6]</td>
<td>73.4/72.2/76.5</td>
</tr>
<tr>
<td>TCEDa</td>
<td>74.2/71.2/75.5</td>
</tr>
<tr>
<td>TCEDb</td>
<td>75.9/71.3/76.2</td>
</tr>
<tr>
<td>TCEDTensorSketch [8]</td>
<td>71.9/70.9/75.1</td>
</tr>
<tr>
<td>TCEDa,LearnableProjection</td>
<td>72.0/68.8/73.4</td>
</tr>
<tr>
<td>TCEDb,LearnableProjection</td>
<td>71.3/68.9/72.6</td>
</tr>
<tr>
<td>TCEDa</td>
<td>74.0/71.0/76.5</td>
</tr>
<tr>
<td>TCEDb</td>
<td>75.6/70.4/76.0</td>
</tr>
</tbody>
</table>

Table 4. The comparison in 50 Salads-eval.

### Table 5

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial CNN [5]</td>
<td>54.4/24.8/32.3</td>
</tr>
<tr>
<td>Spatiotemporal CNN [5]</td>
<td>59.4/45.9/55.9</td>
</tr>
<tr>
<td>IDT+LM [11]</td>
<td>48.7/45.8/44.4</td>
</tr>
<tr>
<td>Dilated TCN [6]</td>
<td>59.3/43.1/52.2</td>
</tr>
<tr>
<td>Bidirectional LSTM [6]</td>
<td>55.7/55.6/62.6</td>
</tr>
<tr>
<td>TCEDmax [6]</td>
<td>64.7/59.8/68.0</td>
</tr>
<tr>
<td>TCEDa</td>
<td>65.8/61.9/68.5</td>
</tr>
<tr>
<td>TCEDb</td>
<td>66.3/62.5/68.9</td>
</tr>
<tr>
<td>TCEDTensorSketch [8]</td>
<td>63.4/62.6/68.5</td>
</tr>
<tr>
<td>TCEDa,LearnableProjection</td>
<td>61.8/58.2/64.4</td>
</tr>
<tr>
<td>TCEDb,LearnableProjection</td>
<td>60.1/56.6/62.9</td>
</tr>
<tr>
<td>TCEDa</td>
<td>64.7/61.3/66.8</td>
</tr>
<tr>
<td>TCEDb</td>
<td>65.7/62.8/69.0</td>
</tr>
</tbody>
</table>

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

### Table 6

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial CNN [5]</td>
<td>74.1/37.7/51.6</td>
</tr>
<tr>
<td>Spatiotemporal CNN [5]</td>
<td>77.9/67.1/77.7</td>
</tr>
<tr>
<td>Spatiotemporal CNN+Seg [5]</td>
<td>74.4/73.7/82.2</td>
</tr>
<tr>
<td>Dilated TCN [6]</td>
<td>78.0/56.8/69.7</td>
</tr>
<tr>
<td>Bidirectional LSTM [6]</td>
<td>74.4/73.7/82.2</td>
</tr>
<tr>
<td>TCEDmax [6]</td>
<td>81.2/85.6/90.3</td>
</tr>
<tr>
<td>TCEDa</td>
<td>82.6/85.6/90.4</td>
</tr>
<tr>
<td>TCEDb</td>
<td>82.2/87.7/91.4</td>
</tr>
<tr>
<td>TCEDTensorSketch [8]</td>
<td>80.8/85.4/90.1</td>
</tr>
<tr>
<td>TCEDa,LearnableProjection</td>
<td>79.7/82.8/88.1</td>
</tr>
<tr>
<td>TCEDb,LearnableProjection</td>
<td>81.6/83.0/89.0</td>
</tr>
<tr>
<td>TCEDa</td>
<td>81.8/85.1/90.0</td>
</tr>
<tr>
<td>TCEDb</td>
<td>81.7/85.1/90.5</td>
</tr>
</tbody>
</table>

Table 6. The comparison in JIGSAWS.
References


[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


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