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{*}
\textsuperscript{2}Max Planck Institute for Intelligent Systems, Tübingen, Germany \textsuperscript{†}
\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 in 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 other 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 with neither information loss nor extra computation. We perform intensive experiments to quantitatively analyze our model and show the superior performances to other state-of-the-art 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 in long-term periods, e.g. daily activities \cite{1}, surgical robots \cite{2}, human motion analysis \cite{3} and animal behavior analysis in the lab \cite{4}. Given a video or a generic time sequence of feature vectors, the action parsing algorithm aims at assigning each frame an action label, so as to partition the entire sequence into several disjoint semantic action primitives. Therefore, it can solve the tasks of action recognition, temporal semantic segmentation and action detection in untrimmed videos under one framework.

In recent years, fine-grained action parsing algorithms based on deep convolutional nets are highly effective. For example, the method proposed in \cite{5} and \cite{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 \cite{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 statistics but different second-order ones. 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 such fact, as well as several recent studies showing that bilinear pooling consistently outperforms first-order pooling on fine-grained tasks (e.g. \cite{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.

However, combining such two methods is highly non-trivial, which requires to overcome drawbacks of the conventional bilinear pooling: (1) The conventional bilinear pooling is designed for visual classification. Thus, the bilinear pooling aggregates the 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 dimension of feature from $d$ to $d^2$, causing parameter proliferation in the neural net and expensive computational cost.

Therefore, in this work we extend the conventional bilinear pooling from several aspects and make it suitable for fine-grained action parsing. Specifically, we have 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 introducing information loss or 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 intensive experiments to investigate our novel bilinear pooling methods, and show that the proposed method consistently improves or 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, for which action segmentation is performed by detecting the state transitions. [11] applies a statistical language model to capture action temporal dynamics. [12] proposes an Ego-ConvNet to perform temporal action segmentation, which incorporates 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 to yield state-of-the-art performances on several fine-grained action datasets. 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.

**Bilinear pooling.** Bilinear pooling (or second-order pooling) is widely used in fine-grained visual classification [7–9, 17–31], visual questioning answering [32–34], feature fusion and disentangling [20, 21, 35–38], action recognition [37–42] and other tasks. In deep neural nets, bilinear pooling is mostly used only once before the classification layer, e.g. in [8, 9, 20, 21, 23, 24, 29, 31, 36, 37], or embedded within the classifier, e.g. in [25, 27].

There are several research directions regarding bilinear pooling: (1) Dimension reduction while minimizing information loss. [8, 30, 39] use tensor sketch [43] to reduce the dimension of vectorized bilinear forms. The studies of [9, 21] use parametric dimension reduction approaches, which can be learned via back-propagation. The work in [33] finds a low-rank approximation of the bilinear forms, so as to convert vector outer product into Hadamard multiplication for cheap computation. [17, 22, 27] utilizes 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. [38] factorizes bilinear composition into consecutive matrix multiplications along different dimensions. [28] uses the low-rank approximation as in [33], and aggregates features hierarchically. (3) Methods to capture richer feature statistics, so that more complex distributions can be represented. [44] propose a higher-order pooling scheme to extract feature co-occurrences of visual words based on linearization of a higher-order polynomial kernel. [45] applies tensor sketch to generate a compact explicit feature map up to p-th order. In spite of increasing the representativeness, more computational loads are introduced. [42] linearizes a Gaussian kernel to propose a higher-order descriptor from the late fusion of CNN classifier scores for the purpose of 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 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 differ-
3 Method

In this section, we first introduce the basic component and the concept of our work, namely, the temporal convolutional encoder-decoder net (TCED) and the conventional bilinear pooling approach. We then present our local bilinear pooling method, where the first-order and the second-order information can be decoupled and can be learned by back-propagation. In the end, we propose a low-dimensional representation and theoretically prove that the proposed representation almost halve the dimensionality without losing information and introducing extra computation.

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 consists of 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, which consists of a 1D temporal convolution layer with an activation function 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. One can refer to [6, Figure 1] for the architecture visualization.

Bilinear Pooling. Given a set of generic feature vectors with \( x \in \mathcal{X} \), such as a set of local spatial image descriptors and a set of frame features, the conventional bilinear pooling \([7, 19–21]\) can be given by

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

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, with the consideration of 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 in order to preserve the temporal structure. Specifically, given a temporal sequence of feature vectors \( X = \{x_1, x_T\} \) with \( x_t \in \mathbb{R}^d \) for \( t \in 1, 2, ..., T \), the local temporal bilinear composition is given by

\[
\mathcal{B}(x_t) = vec \left( \frac{1}{|\mathcal{N}(t)|} \sum_{\tau \in \mathcal{N}(t)} x_{\tau} \otimes x_{\tau} \right),
\]

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

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 indicate different dynamical states, we consider to separate first and second-order components from the bilinear form to describe the action via separate attributes. In this case, we denote the bilinear form in Eq. (2) as \( \mathcal{B}_c(\cdot) \) (meaning coupled) and denote the decoupled bilinear form as \( \mathcal{B}_d(\cdot) \). Provided the feature time sequence \( \{x_1, ..., x_T\} \), the first-order component \( \mu \), the second-order component \( \Sigma \) and \( \mathcal{B}_d(\cdot) \) are given by

\[
\mu_t = \frac{1}{|\mathcal{N}(t)|} \sum_{\tau \in \mathcal{N}(t)} x_{\tau},
\]

\[
\Sigma_t = \frac{1}{|\mathcal{N}(t)|} \sum_{\tau \in \mathcal{N}(t)} (x_{\tau} - \mu_t) \otimes (x_{\tau} - \mu_t)
\]

and

\[
\mathcal{B}_d(x_t) = \left( \mu_t^T, vec(\Sigma_t) \right)^T,
\]

in which one can note \( \mathcal{B}_d(x_t) \in \mathbb{R}^{d(d+1)} \). 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 to describe the statistics is not sufficient. Rather than applying higher-order statistics like in \([44, 45]\), we stay in second-order statistics to retain the computational load low. 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

$$B_c(x_t) = vec \left( \sum_{\tau \in N(t)} \omega_\tau x_\tau \otimes x_\tau \right),$$  \hspace{1cm} (6)

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

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

$$\mu_t = \sum_{\tau \in N(t)} p_\tau x_\tau, \hspace{1cm} (7)$$

$$\Sigma_t = \sum_{\tau \in N(t)} q_\tau (x_\tau - \mu_t) \otimes (x_\tau - \mu_t) \hspace{1cm} (8)$$

$$B_d(x_t) = \left( \mu_t^T, vec(\Sigma_t) \right)^T, \hspace{1cm} (9)$$

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, which can lead to diverging spectra in the bilinear forms and very sparse features before the final classification layer. Here we present three normalization methods that can address such problems.

**l2 normalization.** We 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 effectively eliminates the diverging spectra problem.

**Regularized power normalization.** When using element-wise power normalization, e.g. in \([7,17,22]\), or matrix \([17,22]\) or higher-order tensor \([17]\) spectral power normalization in intermediate layers of a neural net, it can cause gradient explosion during back-propagation, since the gradient goes to infinity when encountering 0 values. Therefore, 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

$$\sigma(x) = RPN(x) = sign(x) \cdot \left( \sqrt{|x| + \theta^2} - \sqrt{\theta^2} \right),$$  \hspace{1cm} (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 refer to \([46]\) for a family of 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

$$\sigma(x) = NReLU(x) = \frac{ReLU(x)}{\max(ReLU(x)) + \epsilon},$$  \hspace{1cm} (11)

where \( NReLU \) 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 \([47]\), \( NReLU \) 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 \( B_c \) and \( B_d \) can capture local temporal statistics which are adaptive to the data. However, the feature dimension is considerably increased. Specifically, given \( x_t \in \mathbb{R}^d \), we have \( B_c(x_t) \in \mathbb{R}^{d^2} \) and \( B_d(x_t) \in \mathbb{R}^{d(d+1)} \). To address such issue, we propose lower-dimensional alternative 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 computational cost as in SVD.

To find such lower-dimensional representations, we first show that \( B_c(\cdot) \) and \( B_d(\cdot) \) are feature mappings associated with reproducing kernel Hilbert spaces (RKHSs) \([48]\), 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 refer to more details in \([49, \text{Chapter 3}]\).

**Proposition 1.** Given \( \{x_1, ..., x_T\} \), we have

$$\langle B_c(x_i), B_c(x_j) \rangle_{\mathbb{R}^{d^2}} = \sum_{\tau \in N(i)} \sum_{\tau' \in N(j)} \omega_\tau \omega_{\tau'} \langle x_\tau, x_{\tau'} \rangle^2_{\mathbb{R}^d},$$  \hspace{1cm} (12)

and
Let \( \langle B_d(x_i), B_d(x_j) \rangle_{\mathbb{R}^{d(d+1)}} = \langle \mu_i, \mu_j \rangle_{\mathbb{R}^d} + \sum_{\tau \in N(i)} \sum_{\tau' \in N(j)} q_{\tau q_{\tau'}} (x_{\tau} - \mu_i, x_{\tau'} - \mu_j)^2 \), \( \langle B_c(x_i), B_c(x_j) \rangle_{\mathbb{R}^d} \)

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

\[ \langle B_c(x_i), B_c(x_j) \rangle_{\mathbb{R}^d} = \left( \sum_{\tau \in N(i)} vec \left( \omega_{\tau} x_{\tau} \otimes x_{\tau} \right), \sum_{\tau' \in N(j)} vec \left( \omega_{\tau'} x_{\tau'} \otimes x_{\tau'} \right) \right) \]

\[ = \sum_{\tau \in N(i)} \sum_{\tau' \in N(j)} \omega_{\tau} \omega_{\tau'} \left( vec \left( x_{\tau} \otimes x_{\tau} \right), vec \left( x_{\tau'} \otimes x_{\tau'} \right) \right) \]

\[ = \sum_{\tau \in N(i)} \sum_{\tau' \in N(j)} \omega_{\tau} \omega_{\tau'} \left( x_{\tau}, x_{\tau'} \right)^2 \]

\[ = \sum_{\tau \in N(i)} \sum_{\tau' \in N(j)} \omega_{\tau} \omega_{\tau'} \left( x_{\tau}, x_{\tau'} \right)^2 \]

\[ = \langle \mu_i, \mu_j \rangle_{\mathbb{R}^d} + \langle vec(\Sigma_i), vec(\Sigma_j) \rangle_{\mathbb{R}^d} \]  

(14)

For the decoupled bilinear composition, we have

\[ \langle B_d(x_i), B_d(x_j) \rangle_{\mathbb{R}^{d(d+1)}} = \langle \mu_i, \mu_j \rangle_{\mathbb{R}^d} + \langle vec(\Sigma_i), vec(\Sigma_j) \rangle_{\mathbb{R}^d} \]  

(15)

Thus, one can easily derive the formula in the proposition.

\[ \square \]

One can see from Proposition 1 that the inner product defined w.r.t. \( B_c(\cdot) \) can be expressed in terms of the 2nd-degree homogeneous polynomial kernel \( k(x, x')^2 \). 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 \( \phi : \mathbb{X} \rightarrow \mathbb{H} \) 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 \( B_c(\cdot) \in \mathbb{R}^d \) be the bilinear composition and \( \phi_c(x) \in \mathbb{R}^{d(d+1)} \) a feature mapping defined by

\[ \phi_c(x) = \left( x_1^2, \ldots, x_d^2, \sqrt{x_1 x_2}, \sqrt{x_1 x_3}, \ldots, \sqrt{x_{d-1} x_d} \right)^T. \]

\[ C \text{ terms} \]

\[ d \text{ terms} \]

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

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

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

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

As discussed before, our proposed bilinear pooling can capture data-adaptive local temporal statistics in the feature time sequence, and the lower-dimensional representations can reduce the model size without sacrificing representativeness. In this section we quantitatively show the effectiveness of our methods. We use the suggested (also the same) architecture setting of the TCED in [6]: The net has two encoders, and the associated two convolution layers have 64 and 96 filters, respectively. The decoders have mirrored structures with the encoders.

4.1 Datasets

In our experiments, the input feature time sequence to the TCED net is extracted from RGB videos using a pretrained VGG-like network [5]. To perform fair comparison with other methods, we downsample the input sequence to achieve the same temporal resolution as [6]. Figure 1 shows several examplar video frames from the three datasets introduced in the following.

**50 Salads** [50]. This multi-modal dataset collects 50 recordings from 25 people preparing 2 mixed salads, and each recording lasts 5-10 minutes. In our experiment we only use the RGB video, which 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, some of which are derived from the high-level actions. For example, the mid-level actions “cut tomato” and “cut cucumber” are derived from the high-level action “cut”. Therefore, we can obtain two subsets 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)** [51] [1]. This dataset contains 7 daily living activities performed
Given the predicted sequence, the edit score is defined as $d_{\text{edit}}(P, Q) \times 100$, in which $d_{\text{edit}}(\cdot, \cdot)$ is the Levenshtein distance. According to the definition, 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 can provide effective evaluation of tasks, in which 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 influenced by over-segmentation results.

**F1 score** [6]. The F1 score is defined as the harmonic mean of the precision and the recall rates of fine-grained action detection, in which the true positive detection is defined as a segment whose action label is same to the ground truth as well as the intersection-over-union of the overlap with the ground truth is greater than 0.1. According to the definition, the F1 score is an evaluation metric from the perspective of action detection, and 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.

**Comparison of different bilinear forms.** In this section, we analyze the influence of the learnable weights used in the coupled bilinear form $(B_c)$ and the decoupled bilinear form $(B_d)$. We denote the corresponding the non-learnable bilinear forms in Eq. (2) and Eq. (3) as $B_{c0}$ and $B_{d0}$, 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 phenomenon is more obvious when the neighbor size is larger. Such result indicates that the learnable weights, i.e. $\{\omega_{\tau}\}, \{p_{\tau}\}$ and $\{q_{\tau}\}$ in equations (6) and (9), enable the derivied 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 individual components are inferior to their combination,
the performances of each ingredient in the decoupled bilinear form are demonstrated in Equation 9. which the first-order component and the second-order component shows the performances of the coupled bilinear form $B_c$ and the decoupled bilinear form $B_d$. (2) The second row shows the performances of each ingredient in the decoupled bilinear form $B_d$, in which the first-order component and the second-order component are demonstrated in Equation 9.

![Figure 2](image)

Figure 2. The performances w.r.t. the neighbor size $|N|$ and the learnability of the weights. From top to bottom: (1) The first row shows the performances of the coupled bilinear form $B_c$ and the decoupled bilinear form $B_d$. (2) The second row shows the performances of each ingredient in the decoupled bilinear form $B_d$, in which the first-order component and the second-order component are demonstrated in Equation 9.

which implies that different orders of information are complementary. This fits our conjecture that first and second-order components tend to describe independent patterns in the data.

**Normalization and activation.** Corresponding to Sec. 3.3, we investigate the influence of normalization and compare different activation functions. In each experiment the neighborhood size of both bilinear forms are identical.

First, different normalization methods are compared in Tab. 1. The neighbor size is identical for both bilinear forms. One can see that $l_2$ normalization and $l_1$ normalization perform almost equivalently, while the normalized ReLU activation function consistently outperforms others. This result can 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 func-

- **NRReLU**
- **NRReLU+l_2**
- **NRReLU+l_1**
- **ReLU+l_2**
- **ReLU+l_1**

Table 1. Different normalization methods are compared, 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>$B_c$</th>
<th>$B_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRReLU</td>
<td>65.82/61.89/68.5</td>
<td>66.28/62.46/68.93</td>
</tr>
<tr>
<td>NRReLU+l_2</td>
<td>64.92/60.01/67.33</td>
<td>66.09/60.02/67.38</td>
</tr>
<tr>
<td>NRReLU+l_1</td>
<td>64.22/60.04/65.86</td>
<td>66.48/63.04/68.71</td>
</tr>
<tr>
<td>ReLU+l_2</td>
<td>64.87/59.88/66.31</td>
<td>64.7/61.45/68.04</td>
</tr>
<tr>
<td>ReLU+l_1</td>
<td>63.05/58.29/65.62</td>
<td>59.76/58.68/64.39</td>
</tr>
</tbody>
</table>

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, neither information loss nor extra computation is introduced. 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 $d(d+1)/2$ for fair comparison. In addition, the LearnableProjection [21] 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. From the tables, one can see 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%. In addition, the low-dimensional representation is on-par, sometime outperforms the corresponding explicit bilinear composition.

4.5 Comparison with State-of-the-art

Here we compare the proposed learnable bilinear forms with the baseline methods. The results of Spatial CNN, Spatiotemporal CNN, CNN+Seg, Dilated TCN, TCN$_{max}$ and Bidirectional LSTM are derived either by [5] and [16], or by ourselves following the studies. More specifically, in Spatial CNN, a linear support vector machine ($C = 1$) is used to classify the frame-wise features directly. In Spatiotemporal CNN, an additional temporal convolution layer with 64 filters and a time-distributed fully connected layer with softmax function are trained jointly to produce parsing results. In Spatiotemporal CNN+Seg, an additional conditional random field is used to yield piecewise constant results.

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\textsubscript{X} denotes the temporal convolutional encoder-decoder with the pooling method \textit{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_B yields 74.7/59.2/66.7 and TCED_max yields 63.6/71.9/75.2 for the GTEA dataset.

5 Conclusion
To our knowledge, we are the first to introduce 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 generalize the averaging operation to convolution with a learnable filter. To reduce dimensionality, we introduce lower-dimensional representations of the bilinear forms, so that neither information loss nor extra computation is introduced. We conduct intensive experiments to analyze the bilinear forms, and show superior performances to state-of-the-art for action parsing. A future work is to investigate higher-order multilinear 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 the German Research Foundation (DFG CRC 1233). All authors faithfully acknowledge Dr. Colin Lea (Facebook, Pittsburgh, PA) to provide frame-wise features of the datasets.

References
[19] Peihua Li, Jianguo Xie, Qilong Wang, and Wangmeng Zuo. Is second-order information helpful for large-scale visual


[27] 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


