



# TSRN: two-stage refinement network for temporal action segmentation

Xiaoyan Tian<sup>1</sup> · Ye Jin<sup>1</sup> · Xianglong Tang<sup>1</sup>

Received: 30 March 2022 / Accepted: 19 April 2023 / Published online: 15 May 2023  
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2023

## Abstract

In high-level video semantic understanding, continuous action segmentation is a challenging task aimed at segmenting an untrimmed video and labeling each segment with predefined labels over time. However, the accuracy of segment predictions is limited by confusing information in video sequences, such as ambiguous frames during action boundaries or over-segmentation errors due to the lack of semantic relations. In this work, we present a two-stage refinement network (TSRN) to improve temporal action segmentation. We first capture global relations over an entire video sequence using a multi-head self-attention mechanism in the novel transformer temporal convolutional network and model temporal relations in each action segment. Then, we introduce a dual-attention spatial pyramid pooling network to fuse features from macro-scale and microscale perspectives, providing more accurate classification results from the initial prediction. In addition, a joint loss function mitigates over-segmentation. Compared with state-of-the-art methods, the proposed TSRN substantially improves temporal action segmentation on three challenging datasets (i.e., 50Salads, Georgia Tech Egocentric Activities, and Breakfast).

**Keywords** Temporal action segmentation · Video semantic understanding · Refinement network · Self-attention · Over-segmentation

## 1 Introduction

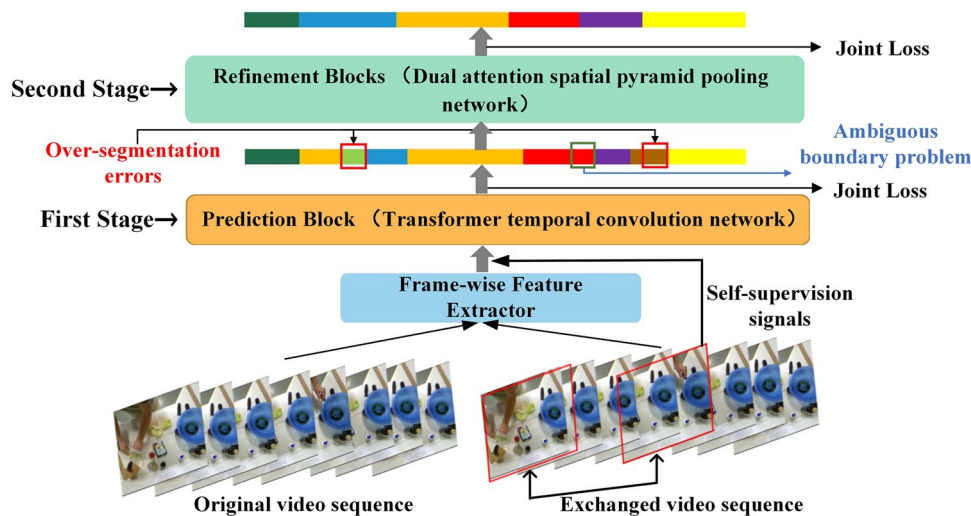
Analyzing and understanding human actions in videos are fundamental to many applications, such as intelligent surveillance [1, 2] and human behavioral analysis [3]. Approaches to recognizing short-trimmed videos to predict action class labels have yielded promising results [4]. However, action segmentation, which aims to assign an action label for each frame to divide the entire video sequence of long untrimmed videos into several disjoint semantic action segments with fine-grained class labels, remains challenging.

Temporal action segmentation is a branch of video-based human action understanding aimed at dividing long untrimmed videos into segment-level snippets and predicting the action labels of frames for the snippets with the same predefined label [5]. Conventional action segmentation methods are based on two-phase deep neural networks: First, two-dimensional (2D) convolutional neural networks [6], two-stream networks [7], or three-dimensional (3D) convolutional neural networks [8] extract low-level spatiotemporal features [48]; Second, using high-level classifiers such as long short-term memory (LSTM) [9] and recurrent neural networks (RNNs) [51] temporally captures frame-wise dependencies. These methods obtain better results on datasets with a small number of action classes than previous methods [43–45]. Nevertheless, they exhibit oscillation predictions on large datasets with various action classes and are usually hard to interpret and correctly train. To tackle these limitations, researchers have suggested a temporal convolutional network (TCN)-based [10] method that captures long-range information using multilayer dilated convolution with increasing an receptive field. The TCN-based method does only model

---

✉ Ye Jin  
jinye@hit.edu.cn  
Xiaoyan Tian  
tianxy@stu.hit.edu.cn  
Xianglong Tang  
tangxl@hit.edu.cn

<sup>1</sup> Harbin Institute of Technology, Harbin 150001, China



**Fig. 1** Structure of the proposed two-stage refinement network (TSRN) comprising a transformer temporal convolutional network (transformer TCN; Sect. 3.1) and a dual-attention spatial pooling network (DASPP; Sect. 3.2). After generating self-supervision signals, the original video sequence and exchanged video

sequence are inputted from a frame-wise feature extractor and fed into the prediction block (transformer TCN) as the first step output of the initial predictions with over-segmentation errors. Then, action segmentation results are refined in the refinement block (DASPP) as the second step

temporal patterns with fewer parameters but is also faster than conventional solutions [9, 51]. It has become a widely used backbone network for the temporal action segmentation task and has led to the development of follow-up works [11–16]. While the progress of recognizing action segments from long untrimmed videos has been made, we still find three technical difficulties worthy of attention in the TCN-based method:

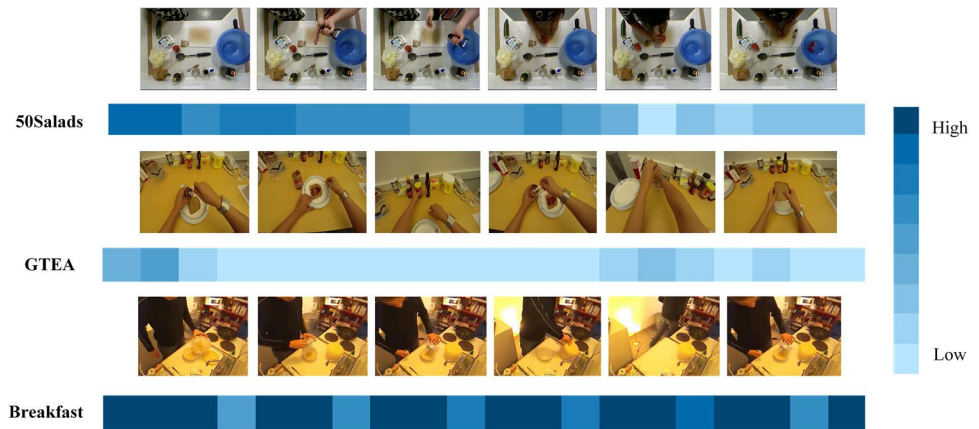
The first is the problem of discounting the semantic association among action segments of an entire video. State-of-the-art methods [11–16] usually adopt a multi-stage architecture based on a TCN that expands temporal receptive fields with dilated convolutions and outputs an initial prediction refined by the subsequent stages to capture long-term relationships. However, the higher dilated convolutional layers lead to the loss of local features and the lack of correlation in long-term dependencies. For instance, Farha et al. [11] simply stacked multiple TCN layers but did not translate this to a corresponding effective performance. This is because focusing only on the features of long-term dependencies tends to lose the semantic association between local action segments, resulting in incorrect predictions for hard-to-recognize frames. In fact, the relations among each action segment indicate a series of continuous activities [16] (e.g., when preparing to make salads, the correct order is adding flavor, cutting vegetables, and mixing). Besides, the underlying contextual information in spatial and channel axes is significant in learning diverse representations in the field of semantic segmentation [52], which motivates us to design a model that can perceive various feature representations of a

video sequence and model different timescales (i.e., short- and long-term timescales) of temporal relationships.

Another difficulty is the inevitable over-segmentation errors [10, 11, 14] for frame-wise classification. In the upper part of Fig. 1, over-segmentation errors occur at the initial predictions when analyzing untrimmed video sequences with a series of actions, which are caused by visual features in one action segment always becoming too similar to those in other action segments. To further reduce over-segmentation errors, recent studies have added additional structures/branches to solve the problem. Wang et al. [12] trained another network to aggregate local predictions by leveraging semantic boundary information but their model incurred higher computational costs. Ishikawa et al. [13] proposed a boundary regression module and used boundary detection on the segmentation outputs for refinement during the post-processing. Moreover, Li et al. [14] and Wang et al. [23] constructed a smooth loss between the log probability of the previous frame and the current frame. However, these methods may incorrectly modify the frame-wise prediction result in the previous predictions and harm the following refinement stages due to the errors in the additional modules or noise in the backbone when identifying ambiguous segment-level action clips.

In addition, the enormous speed and duration variance increase the difficulty of classifying action boundaries. For instance, one “crack\_egg” action is completed in 2 s on the Breakfast dataset [19], but the “fry\_egg” action persists for 2 min. We present examples of the frame-wise variance of I3D features [42] for 21 frames on three challenging datasets (50Salads [17], GTEA [18], and Breakfast [19]) in Fig. 2.

**Fig. 2** Frame-wise variance results of 50Salads, GTEA, and Breakfast datasets



It is obvious that it is the large changes between adjacent action segments, which brings the problem of identifying action boundaries. The ambiguous boundary problem manifests the difficulty of labeling the start or end of an action segment (seen in Fig. 1), which should be solved to understand untrimmed videos.

Inspired by [53], where objects were detected in a multiscale vision range, we introduce a two-stage refinement network (TSRN) that captures macroscale and microscale features to solve the difficulties mentioned. In Fig. 1, the proposed TSRN consists of a frame-wise feature extractor and two stages: a transformer temporal convolutional network (transformer TCN) and a dual-attention spatial pyramid pooling network (DASPP). Unlike general models that use the same subnetworks to expand temporal receptive fields by boosting the network depth, the proposed TSRN redefines the architecture and modifies the meaning of each stage.

For the transformer TCN, the transformer encoder block is designed to explore the global features of a video sequence effectively and then to use multiple dilated convolutional layers to model the long-range temporal dependency. To refine the initial predictions from the first stage, we regard the DASPP as the second stage, which eliminates over-segmentation errors from the initial predictions by understanding the video sequence's global and local context information, thus producing more accurate predictions of action boundaries. For DASPP, a channel attention module (CAM) is proposed to capture channel context via reallocating weights with the importance of channels, a spatial attention module (SAM) is aimed at generating attention weights to adapt the most informative video parts, and a spatial pyramid pooling module (SPP) is used to integrate multiscale features of a video sequence. Furthermore, self-supervised signals simulate over-segmentation errors to locate wrong temporal ordered frames and revise them in the predictions. For model training, to force the TSRN to correct mislabeled frames in the previous predictions, we form a joint loss to combine the auxiliary self-supervised function, a traditional loss function

[11, 13, 14], and a focal loss function that smooths the transition of action probabilities predictions. The contributions of this study can be summarized as follows:

1. We design the novel TSRN that adopts a two-stage strategy to capture macroscale and microscale features from video sequences. The TSRN comprises a transformer TCN and a DASPP to overcome the technical difficulties above, improving single-model classification results by up to 22.8% for the F1 score and 13.3% in terms of segmental edit distance.
2. A transformer TCN is proposed to model global dependency by exploring the correlations among frames, and DASPP is adapted to combine a video's global and local features. To our best knowledge, it is the first attempt to leverage channel and spatial attention information for temporal action segmentation.
3. We introduce a joint loss function to smooth the transition of action probabilities and experiment with the combination of loss functions for our model. Combining an auxiliary self-supervised function and a focal loss function provides a 12.8% improvement in the F1 score and an 11.4% improvement in segmental edit distance.
4. The proposed TSRN achieves state-of-the-art performance on three challenging benchmarks for temporal action segmentation: 50Salads [17], Georgia Tech Ego-centric Activities (GTEA) [18], and Breakfast [19].

## 2 Related work

### 2.1 Action segmentation

Action segmentation aims to segment a video sequence according to the semantic meaning and label each segment-level action corresponding with predefined labels temporally. In earlier approaches, a sliding window method [46, 47] with non-maximum suppression is used to detect action

segments. Other traditional methods use Markov models [29, 40] on the top of frame-wise classifiers. However, these approaches are very slow and exist to solve the maximization problem over long sequences.

Inspired by the success of speech synthesis, researchers have proposed diverse temporal convolutional networks (TCNs) from WaveNet [20]. Lea et al. [10] proposed the encoder-decoder TCN (ED-TCN), [21, 50] expanded it to the temporal deformable residual network with a residual stream to analyze video information. Although these approaches obtained long-range dependencies, the increasing pooling and upsampling operations might discard the fine-grained details of video sequences. To overcome these challenges, a multi-stage TCN (MS-TCN) [11] was designed, which used dilated 1D convolutions to enlarge temporal receptive fields instead of pooling operations in [10] with a full resolution. Based on [11], dilated TCNs [11, 14, 22, 23, 49] and a temporal reasoning module with graph convolutional networks [16, 24] can be fed into the top of temporal action segmentation models, modeling on the full resolution, capturing long-range dependencies, and learning fine-grained features of video sequences. Other works, such as [12, 13], are based on the anchor-free temporal action proposal task, distinguishing actions or the possibility of judging whether a frame starts or ends. Wang et al. [12] trained an extra network to smoothen action boundaries, and Ishikawa et al. [13] used an action boundary regression network to mitigate over-segmentation errors by detecting action boundaries. However, training a large and time-consuming model limited the performance. Recently, in [15, 25], domain adaption was introduced to the action segmentation task. Gao et al. [26] used hierarchical artificial design receptive fields to build segmentation models, but they neglected the importance of global and local contexts of the whole video sequence.

In this study, our model is based on dilated TCNs and uses a two-stage architecture to capture different timescale features of video sequences and generates smooth predictions over segment-level action boundaries with low computation costs.

## 2.2 Transformer

The transformer [32] is initially applied for natural language processing tasks. With the immense potential of machine translation and English constituency parsing [27], researchers have recently grown a great interest in applying transformer-based models for computer vision tasks, such as object detection [28], image classification [30], and segmentation [31]. Considering that a transformer is inherently well suited for sequence-based tasks, we attempt to incorporate the transformer-based models into the action segmentation task, which models relations among segment-level actions of a video sequence. Note that the self-attention mechanism

[32] is the fundamental component in the transformer-based models. This mechanism precisely computes the output at each position of video sequences by calculating attention scores for all positions and fusing the intrinsic features based on the scores. While a single-head self-attention layer only focuses on more meaningful position information, multi-head attention allows the model to gather information from different representation subspaces. Our model utilizes a multi-head self-attention mechanism to enhance the semantic association among the local action segments and model temporal long-range dependencies in videos.

## 2.3 Attention mechanism

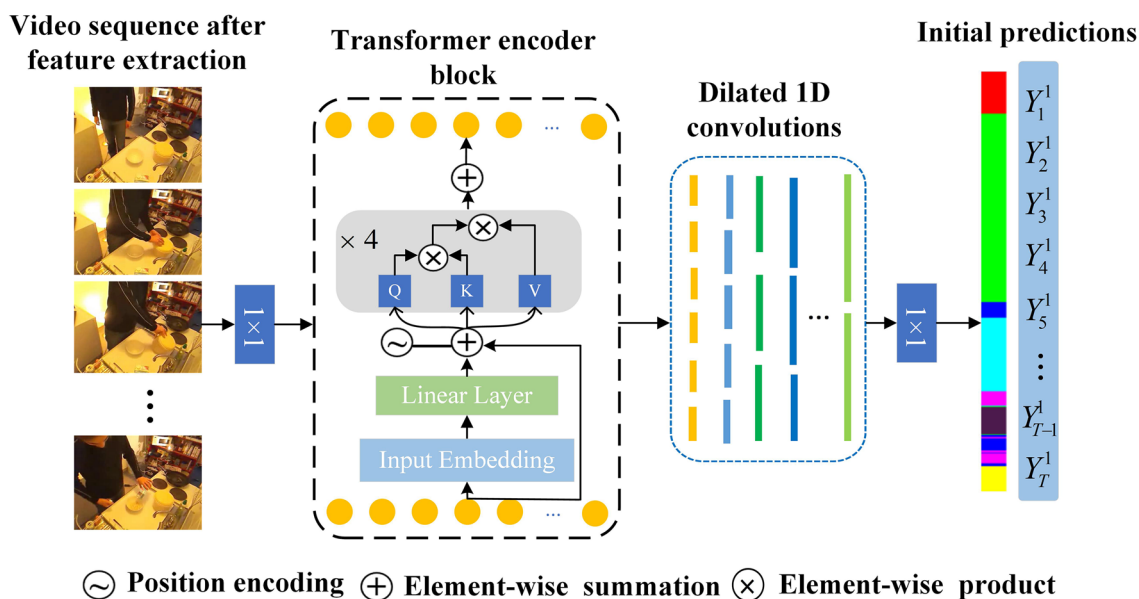
The attention mechanism plays a vital role in analyzing and understanding complex scenes [33, 34], diverting attention to the most critical parts of an image and taking no notice of irrelevant regions. Some extensive research works on this domain are relevant to our work. For example, Hu et al. [35] proposed the sequence-and-excitation (SE) module to explore the inter-channel relationship and automatically learn the effectiveness of different channel-wise attentions. Based on the SE module, Woo et al. [36] introduced the spatial attention mechanism, which considers that max-pooling operation makes a network pay attention to the essential channel-wise features, focusing on target space areas. Some researchers have recently studied the potential for transformer-based models in image processing and proposed the vision transformer (ViT) [30] as a pure attention-based network with a multi-head attention core. Motivated by the above attention mechanism, we introduce the dual-attention mechanism with a spatial pyramid pooling module (SPP) [39] to explore the applications for the action segmentation task.

In addition to the attention-based module for adaptive feature refinement in the previous works [35, 36, 39], we extract multiscale features of the video sequence with different receptive fields and fuse them in the channel dimension of the feature maps to develop features with minimal modifications. Finally, we eliminate incorrect frame-wise predictions by focusing on adjacent action segments from a local perspective and reduce over-segmentation errors from the prediction block by fusing long- and short-term features.

## 3 Approach

This section introduces the proposed temporal action segmentation approach, i.e., TSRN. Our structure consists of a frame-wise feature extractor and two networks, a transformer TCN and a DASPP, as shown in Fig. 1. The frame-wise feature extractor takes original frames and exchanged frames as the input, generating input features as input. Then,





**Fig. 3** Overview of the transformer temporal convolutional network (transformer TCN). The transformer TCN contains a transformer encoder block and several dilated 1D convolutions using residual connections

the transformer TCN develops initial predictions in the first stage. This stage adapts a multi-head self-attention mechanism with several dilated 1D convolutions. In the second stage, the TSRN revises previous predictions from the prediction block by stacking refinement blocks that involve a dual-attention model, an SPP module, and dilated residual layers.

The remainder of the section is organized as follows: Sect. 3.1 illustrates how the transformer TCN models long-range dependencies and develops initial predictions. Section 3.2 introduces the multiscale features fusion module in the DASPP to revise predictions, such as over-segmentation errors. Finally, Sect. 3.3 describes how the joint loss forms, and Sect. 3.4 details the experimental setup.

Let  $X_{1:T} = (X_1, \dots, X_T) \in R^{T \times D_{dim}}$  and  $X_{1:T}^{ex} = (\dots, X_{i_1}, \dots, X_{i_2}, \dots) \in R^{T \times D_{dim}}$  be the inputs to the TSRN, where  $T$  is the number of frames in a video and  $D_{dim}$  is the feature dimension. Our goal is to classify the frame-wise action class  $C_{1:T} = (C_1, \dots, C_T)$ , whose ground-truth label is set as  $Y_{1:T}^{gt} = (Y_1^{gt}, \dots, Y_T^{gt})$ , where  $Y_t^{gt} \in \{0, 1\}^C$  is a one-hot vector representation of whether the  $i$  th frame is predicted as the true label,  $C$  is the number of action classes,  $X_{1:T}^{ex}$  is swapped in pairs and formed with the wrong temporal order.

### 3.1 Transformer TCN

As shown in Fig. 3, the first layer of the transformer TCN is a  $1 \times 1$  convolutional layer that adjusts the dimension of input features to match the number of feature maps  $D$ . Then,

a transformer encoder block is included with the multi-head self-attention mechanism, and its output is transferred to several layers of dilated 1D convolutions with a kernel size of 3. Subsequently, a  $1 \times 1$  convolutional layer is applied after the output of the last dilated 1D convolutional layer, followed by a softmax activation to get the action class probabilities as the first-stage predictions  $Y^1 = (Y_1^1, \dots, Y_T^1)$ .

Concretely, a sinusoidal positional encoding module [32] with dimension  $D$  is first incorporated into the original embedding in the transformer encoder block to form the input vector  $I \in R^{T \times D}$ . Second, for the input vector  $I$  and the number of heads  $h$ , the input vector is transformed into three representative groups (i.e., the query group, the key group, and the value group). In each group, there are  $h$  vectors with dimensions  $d_q = d_k = d_v = D/h$ . Vectors derived from different groups are then packed together into three different groups of matrices:  $\{Q_i\}_{i=1}^h$ ,  $\{K_i\}_{i=1}^h$ , and  $\{V_i\}_{i=1}^h$ . Formally, the multi-head self-attention process is shown as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^o, \tag{1}$$

where  $\text{head}_i = \text{ATTN}(Q_i, K_i, V_i)$ ,

where *MultiHead* represents multi-head self-attention, *Concat* denotes the concatenation operation, *ATTN* indicates the attention mechanism;  $Q$ ,  $K$ , and  $V$  are the concatenation of  $\{Q_i\}_{i=1}^h$ ,  $\{K_i\}_{i=1}^h$ , and  $\{V_i\}_{i=1}^h$  respectively. Here, we set  $h=4$  ( $h=2$  for the GTEA and Breakfast datasets) for the number of heads. To facilitate residual connections, a feed-forward network is applied after the multi-head self-attention layer, which consists of an embedding layer and a linear layer,

producing the output of the transformer encoder block, which can be formulated as

$$Y_{\text{trans-block}} = W_1(W_2(\text{MultiHead}(Q, K, V))), \tag{2}$$

where  $W_1$  and  $W_2$  are the two parameter matrices of two linear transformation layers.

After focusing on the semantic association among the action segments in the transformer encoder block, we need to extract different receptive fields of temporal features to capture the global information. In dilated 1D convolutions, each layer applies  $D$  dilated convolutions with rectified linear unit activation and a convolutional layer. We further use residual connections to facilitate the gradient flow. The output of each dilated residual layer  $k \in \{1, 2, \dots, K\}$  as  $L_k \in R^{T \times D}$  can be described by

$$L_k = L_{k-1} + W_4 \otimes (\text{ReLU}(W_3 \otimes L_{k-1} + b_1)) + b_2, \tag{3}$$

where  $L_{k-1}$  is the output of the  $(k - 1)$ th dilated residual layer,  $W_3 \in R^{3 \times D \times D}$  denotes the weight matrix of dilated 1D convolution filters with a kernel size of 3,  $D$  is the number of dilated convolutional filters,  $W_4 \in R^{1 \times D \times D}$  is the weight of the  $1 \times 1$  convolution, and  $b_1, b_2 \in R^{T \times D}$  are bias vectors. To capture long-range dependencies of the video sequence, we follow [11] that stacked dilated residual layers to expand receptive fields. Because the receptive fields exponentially grow with the number of layers, we obtain large receptive fields with few layers, thus inhibiting over-fitting during the training of the model. The receptive field of each layer can be formulated as

$$\text{Receptive Field}(k) = 2^k - 1, \tag{4}$$

where  $k \in [1, K]$  denotes the layer number. Followed by the last dilated residual layer  $K$ , we apply a  $1 \times 1$  convolution after a softmax activation layer, i.e.,

$$Y^1 = \text{Soft max}(W \otimes L_K + b), \tag{5}$$

where  $Y^1 = (Y^1_1, \dots, Y^1_T) \in R^{T \times C}$  represents the action class probabilities at time  $t$  for the first-stage prediction of the TSRN,  $W \in R^{C \times D}$  and  $b \in R^C$  are the weight and bias of the  $1 \times 1$  convolutional layer, respectively,  $C$  is the number of action classes.

Different from the MS-TCN [11], which stacks some single-stage TCNs [11] and leads to the loss of local information in higher layers, we use the transformer TCN to extract the frame-wise features and generate the first-stage prediction. To obtain the long-range dependencies of the entire video, we utilize a transformer-based model that models temporal relations to generate the local features among action segments and then constantly perceives the global features of the whole video.

### 3.2 DASPP

Although the transformer TCN has improved action segmentation predictions, the results still include over-segmentation errors. Recent methods [11, 14] have focused on modeling different timescales of features by stacking additional layers that might lose the local information of a video sequence. Nevertheless, we use DASPP to revise the initial prediction estimated in the first stage and then selectively aggregate local and global features by employing the multi-stage architecture.

Given  $Y^1$ , DASPP aims to refine the first-stage prediction by fusing multiscale features and revising segment-level action boundaries, alleviating over-segmentation errors. In DASPP, each refinement block takes predictions from the previous block and then refines them. The input of each refinement block in DASPP is

$$Y^1 = Y^1_1, \dots, Y^1_T, \tag{6}$$

$$Y^i = F(Y^{i-1}) \quad i \in [2, 5],$$

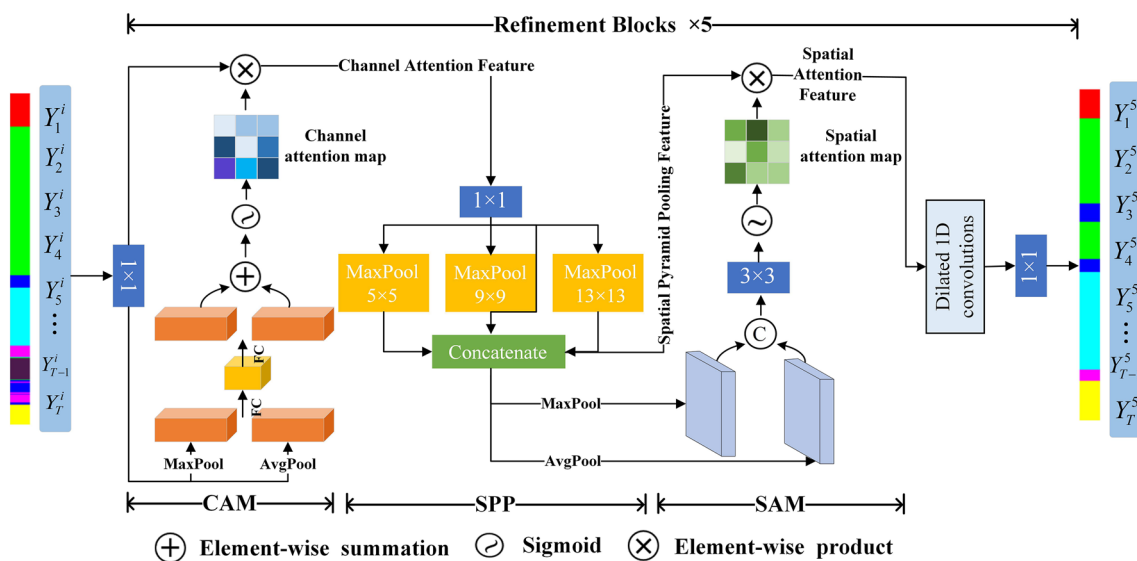
where  $Y^1$  is the input of the first refinement block,  $Y^i$  is the output of block  $i$ , and  $F(\cdot)$  shows the multi-stage operation of DASPP. As shown in Fig. 4, each refinement block of the DASPP consists of a channel attention module (CAM), a spatial pyramid pooling module (SPP), a spatial attention module (SAM), and 10 dilated 1D convolutional residual layers with a kernel size of 3. To receive the probabilities for the output class  $Y^5 = (Y^5_1, \dots, Y^5_T) \in R^{T \times C}$  as the second-stage refinement prediction, we apply a  $1 \times 1$  convolutional layer after the last dilated residual layer, followed by a softmax activation layer in each refinement block.

CAM Channel attention is widely utilized to distinguish the significance of different channels, thus strengthening meaningful channel features, and suppressing redundant features in computer vision. We propose a CAM for a feature representation sequence to capture channel context via reallocating weights with the importance of channels. To the left in Fig. 4, given a feature representation as input, CAM reduces the number of channels to learn the local dependency between channels via max pooling and average pooling operations. Then, CAM increases the number of channels returning to the original size and produces the channel attention map  $M_c$  by the sigmoid activation. The CAM process of the  $i$ th refinement block in DASPP can be summarized as

$$M_c(Y^i) = \text{Sigmoid}(W_5(W_6(Y^i_{c-\text{avg}(t)})) + W_5(W_6(Y^i_{c-\text{max}(t)}))), \tag{7}$$

$$Y^i_{c(t)} = Y^i \otimes M_c(Y^i), \tag{8}$$

where  $Y^i_{c-\text{max}(t)}$  and  $Y^i_{c-\text{avg}(t)}$  are max pooling and average pooling descriptors in a multilayer perceptron network,  $W_5$  and  $W_6$  are the weights of the 1D convolution with a kernel



**Fig. 4** Dual-attention spatial pyramid pooling network (DASPP) contains five refinement blocks. Each refinement block includes a channel attention module (CAM), a spatial pyramid pooling module (SPP), a spatial attention module (SAM), and several dilated 1D convolutions

size of 1,  $\otimes$  denotes the element-wise product. Equation (7) indicates the process of the channel attention map with a dimension-reduction operation, and Eq. (8) is the output of the channel attention mechanism by the dimension-increasing function.

**SPP** Similar to a feature pyramid network (FPN) [37], the SPP relies on the pyramidal shape of the feature hierarchy to extract multiscale features with strong semantics. Although the transformer TCN has perceived the global information of a video sequence, variable pooling kernels can supplement the local context from the input sequence. In SPP, we overcome the shortcoming of losing fine-grained information with limited receptive fields by combining multiscale features with a large temporal receptive field to refine the prediction.

Our SPP is composed of four parallel max-pooling layers with kernels of  $1 \times 1$ ,  $5 \times 5$ ,  $9 \times 9$ , and  $13 \times 13$ , which extract multiscale features and fuses them by concatenating them in the channel dimension of feature maps. As the multiscale features obtained by SPP are expected to refine the predictions with a small computation cost, the lightweight module can be integrated into the DASPP effectively. SPP of the  $i$ th refinement block is computed as

$$Y_{spp(t)}^i = Y_{c(t)}^i + \text{Concat}(f^{1 \times 1}(Y_{c(t)}^i); f^{5 \times 5}(Y_{c(t)}^i); f^{9 \times 9}(Y_{c(t)}^i); f^{13 \times 13}(Y_{c(t)}^i)), \tag{9}$$

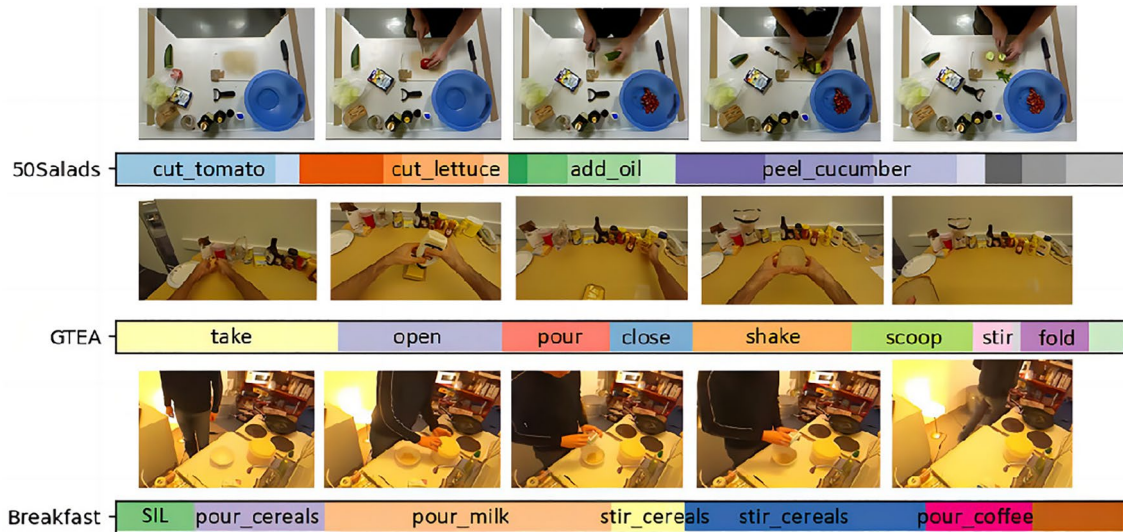
where  $Y_{spp(t)}^i = (Y_{spp(1)}^i, \dots, Y_{spp(T)}^i) \in R^{T \times D}$ . Moreover,  $f^{1 \times 1}$ ,  $f^{5 \times 5}$ ,  $f^{9 \times 9}$ , and  $f^{13 \times 13}$  represent pooling operations with the filters of  $1 \times 1$ ,  $5 \times 5$ ,  $9 \times 9$ , and  $13 \times 13$ , respectively.

**SAM.** It is acknowledged that common untrimmed video samples last for 2–3 min, and the samples are downsampled at a rate of 15 or 30 per second, which is difficult to distinguish the most worthy information across the frames. Under limited computing resources, it is necessary to allocate resources for the most informative part of frames in a video. Apart from CAM, which assigns appropriate weights according to the importance of the channels, SAM focuses on identifying different classifications of adjacent action segments and combining features along the channel axis.

As shown in the right of Fig. 4, SAM first compresses the dimension of input features from  $T \times D$  to  $T \times 1$  through average-pooling and maximum-pooling operations. The corresponding feature descriptors  $Y_{s-avg(t)}^i$  and  $Y_{s-max(t)}^i$  are processed by a  $3 \times 3$  convolutional layer to squeeze multichannel features into a single channel, generating a spatial attention map  $M_s$ :

$$M_s(Y_{spp(t)}^i) = \text{Sigmoid}(f^{3 \times 3}(\text{Concat}(Y_{s-avg(t)}^i; Y_{s-max(t)}^i))), \tag{10}$$

where  $f^{3 \times 3}$  denotes a convolution operation with a filter size of  $3 \times 3$ . The relations among adjacent frames are captured by Eq. (10) to represent the information importance in each frame. Then, we multiply the spatial attention map  $M_s$  with the intermediate features  $Y_{spp(t)}^i$  to generate spatial features  $Y_{s(t)}^i = (Y_{s(1)}^i, \dots, Y_{s(T)}^i) \in R^{T \times D}$ . The SAM process of the  $i$ th refinement block is



**Fig. 5** Distribution of action classes for the 50Salads, GTEA, and Breakfast datasets

$$Y_{s(t)}^i = Y_{spp(t)}^i \otimes M_s(Y_{spp(t)}^i), \quad (11)$$

where  $\otimes$  denotes an element-wise product.

### 3.3 Joint loss

To train TSRN, we use the loss function as in Farha and Gall [11], which comprises the cross-entropy loss  $L_{cls}$  and the regularization loss  $L_{reg}$  for classifying and smoothing each frame. In addition, the proposed focal loss  $L_{focal}$  solves the imbalanced frequency among action classes during training. In Fig. 5, we have illustrated the calculated number of instances per action class for the three datasets, and the results indicate a significant class imbalance. If there are no reasoning weighting restrictions, an imbalance training may cause over-segmentation errors. Moreover, the auxiliary self-supervised loss  $L_{self}$  (including  $L_{ex}$  and  $L_{corr}$  [16]) enhances temporal reasoning by exchanging frames in pairs and strengthens the connection between short- and long-term timescales. Thus, it identifies exchanged frames and predicts the correct action labels at their corresponding instances.

#### 3.3.1 Classification loss

We adopt the cross-entropy loss  $L_{cls}$  to determine the proximity between the prediction and ground truth:

$$L_{cls} = \frac{1}{T} \sum_t \sum_c -Y_{t,c}^{i(gt)} \log(Y_{t,c}^i) = \frac{1}{T} \sum_t -\log(Y_{t,c}^i), \quad (12)$$

where  $Y_{t,c}^i$  is the predicted probability for the target label  $c$  at time  $t$  of the  $i$ th block in our TSRN, and  $Y_{t,c}^{i(gt)}$  is the ground-truth label corresponding to  $Y_{t,c}^i$ .

#### 3.3.2 Regularization loss

While the classification loss treats each frame independently, it might cause over-segmentation errors. To encourage smooth transitions between frames, we use the truncated mean squared error proposed in [11] as the regularization loss:

$$L_{reg} = \frac{\lambda}{TC} \sum_c \sum_t (Y_{t-1,c}^i - Y_{t,c}^i)^2, \quad (13)$$

where  $T$  is the length of the video,  $C$  is the number of action classes, and  $\lambda$  denotes the hyperparameter sets to 0.15.

#### 3.3.3 Focal loss

In multi-class classification, a balanced dataset has target labels that are evenly distributed. In real scenarios, datasets usually have an imbalanced distribution of action instances, which may cause two problems: (1) Most instances are defined as well-classified samples that contribute no meaningful training information. (2) The well-classified samples might overwhelm the training and lead to model degradation. The frequency of different action segments varies for each action class, which results in imbalanced weightings during training. Thus, we impose the focal loss [38] to down-weight the well-classified samples such that their contribution to the joint loss is small, even though the amount of their samples is large, and focuses on the hard-to-classify samples. The focal loss function is defined as follows:



$$L_{focal} = -\alpha(1 - Y_{t,c}^i)^\gamma \log(Y_{t,c}^i), \tag{14}$$

where  $\alpha$  is the weighting factor for balancing the weights of all action classes, and  $(1 - Y_{t,c}^i)^\gamma$  is the modulating factor with the focusing parameter  $\gamma$ , which focuses on hard-to-classify samples during training by reducing the weights of the well-classified samples among different action segments.

### 3.3.4 Auxiliary self-supervised loss

Due to the inherent temporal information of videos that can be used as supervision signals for self-supervised auxiliary tasks, we follow [16] that simulated over-segmentation errors in the temporal action segmentation results to bolster the temporal relations among action segments.

We select 20% of frames in pairs from the input video sequence  $X_{1:T}$  and exchange them into the wrong temporal order  $X_{1:T}^{ex}$ . The output corresponds to  $X_{1:T}^{ex}$  containing action likelihoods  $Y_{1:T}^{i(ex)} \in R^{T \times C}$  and exchanged likelihoods  $e_{1:T}^{i(ex)} \in R^{T \times 2}$ . Besides, binary self-supervised signals  $p_{1:T} = (p_1, \dots, p_T)$  are designed to label the frames, where  $p_i = \{0, 1\}^2$  is the one-hot vector representing whether the  $i$  th frame is exchanged. Based on the temporal order information obtaining an absolute dominance for simulating over-segmentation errors, the original video sequence’s ground-truth label  $Y_{1:T}^{(gt)} \in \{0, 1\}^C$  and auxiliary self-supervised signals are used as training labels for the auxiliary self-supervised loss. The auxiliary self-supervised loss is

$$L_{self} = L_{ex}(e_{1:T}^{i(ex)}, p) + L_{corr}(Y_{1:T}^{i(ex)}, Y_{1:T}^{(gt)}), \tag{15}$$

The output of our network is frame-wise action predictions. Therefore, the final loss function to train the TSRN is the combination of the four losses

$$L = \sum_i L_{cls} + L_{reg} + L_{focal} + L_{self}, \tag{16}$$

**Table 1** Details of the datasets used in this study. #classes and #videos are the number of action classes and videos, respectively. #instances/video is the average number of action classes of each video

	# classes	# videos	# instances/video	# cross-validation	Description
50Salads	17	50	20	5	Prepare for salads
GTEA	11	28	20	4	Daily activities
Breakfast	48	1712	6	4	Making breakfast

where  $i$  is the number of the blocks in TSRN ( $i = 6$  and includes one prediction block and five refinement blocks.)

### 3.4 Experimental setup

The TSRN consists of two stages: 1) a prediction block and 2) five refinement blocks. We use 64 convolutional filters (128 for the GTEA dataset) for all blocks, and the kernel size is 3. Because the GTEA dataset contains the fewest action classes and videos of the datasets listed in Table 1, more features are required to classify the frames of action segments during model training. For the transformer TCN, we set the number of dilated residual layers to 11 ( $K = 11$ ). For the DASPP, we set the number of layers to 10 ( $L = 10$ ). In addition, in focal loss  $L_{focal}$ , we keep  $\gamma = 2$  for all datasets,  $\alpha = 0.15$  for the 50Salads and Breakfast datasets, and  $\alpha = 0.25$  for the GTEA dataset. We train the model for 100 epochs in all experiments using Adam optimization with a learning rate of 0.0005 and a batch size of one [11, 13, 14]. During network training, action segmentation results from the transformer TCN are predictions refined by the DASPP. Our implementations are based on the PyTorch library and implemented on a computer equipped with an NVIDIA TESLA V100 graphics processor.

## 4 Experiments

In this section, we describe the datasets and evaluation metrics. Then, we report the ablation studies and their results. Finally, we compare the proposed TSRN with the state-of-the-art temporal action segmentation methods and provide qualitative results.

### 4.1 Datasets and metrics

**Datasets** We evaluate our TSRN on three challenging datasets: 50Salads [17], GTEA [18], and Breakfast [19]. Table 1 shows the details of the three challenging datasets. The 50Salads dataset contains over four hours of annotated accelerometer data and 50 RGB-D videos and captures 25 actors preparing to mix two different salads. On average, each video consists of 20 action categories and keeps 6.4 min. For evaluation, we use five-fold cross-validation and count the average value as the final results. The GTEA dataset contains 28 egocentric videos and seven daily activities, such as taking, pouring, and opening, each performed by four different subjects. We follow four-fold cross-validation as prior works. The Breakfast dataset is the largest dataset with 1712 videos, which comprises 48 different action classes related to breakfast preparation, performed by 52 different individuals in 18 different kitchens, and each video has six

**Table 2** Comparison between one- and two-stage architectures on the 50Salads dataset

	F1@{10, 25, 50}			Edit	Acc
One stage w/ the transformer TCN	61.2	58.9	53.3	54.1	83.4
One stage w/ the DASPP	76.0	73.3	65.2	68.7	78.8
<b>Two-stage TSRN</b>	<b>84.9</b>	<b>83.5</b>	<b>77.3</b>	<b>79.3</b>	<b>84.5</b>

The bold values indicate the highest value for each metric (column) in that table/comparison

action categories on average. We follow [19] for the evaluation, suggesting the standard four-split cross-validation.

For the three datasets, we follow [10–12, 14, 16, 23, 26] that extracted I3D [42] features for the video sequences and use these features as the input to our model in all experiments. For each frame, the feature is obtained by concatenating the RGB and flow streams, which means the dimensions of the pre-extracted feature sequences are  $T \times 2048$ .

**Metrics.** We report the three metrics employed in [11] for the above datasets, namely, frame-wise accuracy (Acc), segmental edit distance (Edit) [41], and F1 score at the IOU thresholds 10%, 25%, and 50%, denoted as  $F1@{10, 25, 50}$  [10]. While *Acc* is the most prevalent metric in deep learning, it is oblivious to the continuity of action segments in the video sequence, which brings about over-segmentation errors in the action segmentation. In addition, large action duration variance in the datasets has an important influence on *Acc*, making this metric unsuitable for measuring the qualitative differences among long action segments. Hence, *Edit* is used to calculate the Levenshtein distance [41] between predictions and ground-truth labels to address this limitation. Meanwhile, the F1 score with the overlapping threshold  $k\%$  ( $F1@k$ ) is defined as  $F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$  to evaluate the quality of the predictions as proposed by [10], where precision and recall are computed for the true positives, false positives, and false negatives summed over all action classes. Similarly, the F1 score also penalizes over-segmentation errors and disregards temporal shifts between

**Table 3** Comparison with or without the multi-head self-attention mechanism on the 50Salads, GTEA, and Breakfast datasets

50Salads	F1@{10, 25, 50}			Edit	Acc
w/o multi-head self-attention	77.3	75.7	67.1	70.2	81.3
<b>w/ multi-head self-attention (ours)</b>	<b>84.9</b>	<b>83.5</b>	<b>77.3</b>	<b>79.3</b>	<b>84.5</b>
GTEA					
w/o multi-head self-attention	85.8	84.4	73.8	80.0	78.2
<b>w/ multi-head self-attention (ours)</b>	<b>89.4</b>	<b>87.8</b>	<b>80.1</b>	<b>84.9</b>	<b>80.6</b>
Breakfast					
w/o multi-head self-attention	49.4	44.7	34.9	55.8	68.0
<b>w/ multi-head self-attention (ours)</b>	<b>75.4</b>	<b>70.3</b>	<b>56.2</b>	<b>75.0</b>	<b>71.6</b>

The bold values indicate the highest value for each metric (column) in that table/comparison

the predictions and ground truth in the temporal action segmentation task.

## 4.2 Evaluation of the two-stage architecture

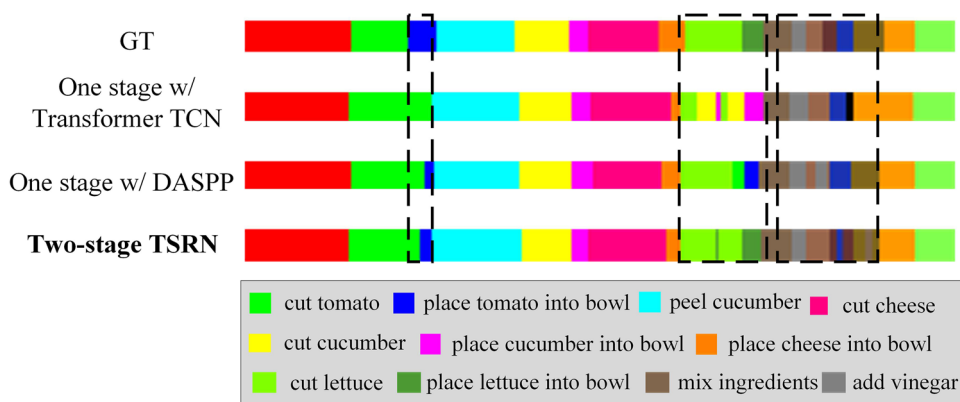
This subsection adds the transformer TCN and DASPP for the prediction and refinement in our two-stage architecture. Table 2 shows that this architecture outperforms the one-stage variants by 24.6% in the F1 score, 25.2% in the segmental edit distance, and 5.7% in the frame-wise accuracy. This highlights the gains of the transformer TCN and DASPP. To determine the impact of utilizing the transformer TCN and DASPP in all stages, we also trained the TSRN with the transformer TCN and DASPP in the two stages. As shown in Table 2, the substantial improvement of TSRN with a two-stage architecture indicates that extracting and moving the refinement part so that it comes after the initial prediction part is critical for the design. While the temporal relations are modeled to access the global features by the transformer TCN, the refinement blocks in the second stage focus on fusing the global and local features using DASPP. Regardless of whether the transformer TCN and DASPP are used in a one-stage architecture, the evaluation metrics drop substantially because of overfitting during training.

Figure 6 shows the qualitative results among several architectures with different color codes. The given video is obtained from the 50Salads dataset, which depicts fine-grained actions of making salads. The segmentation results show the one-stage architecture with the transformer TCN or DASPP wrongly classifies “cut\_tomato” as “place\_tomato\_into\_bowl,” “cut\_lettuce” as “cut\_cucumber,” “place\_cucumber\_into\_bowl,” and “cut\_tomato.” Our two-stage TSRN indicates that the model can infer activities around neighboring action segments in global semantic relationships (e.g., “The process of dealing with lettuce is continuous, which makes it asemantic to predict extra actions, such as placing tomato into the bowl.”). Moreover, ambiguous frames near the action boundaries have been alleviated based on the two-stage TSRN, shown in the black boxes in Fig. 6. Therefore, our two-stage TSRN mitigates over-segmentation

**Table 4** Performance of the multi-head self-attention mechanism with different heads and layers on the 50Salads dataset

	F1@{10, 25, 50}			Edit	Acc
one head	83.1	80.1	70.3	76.1	84.0
<b>two heads</b>	<b>84.9</b>	<b>83.5</b>	<b>77.3</b>	<b>79.3</b>	<b>84.5</b>
four heads	82.7	80.4	73.2	75.0	83.0
one layer	80.1	78.3	69.5	75.3	82.0
two layers	81.1	80.2	70.6	75.1	82.1
<b>four layers</b>	<b>84.5</b>	<b>83.4</b>	<b>77.3</b>	<b>79.3</b>	<b>84.5</b>

The bold values indicate the highest value for each metric (column) in that table/comparison



**Fig. 6** Qualitative results of the temporal action segmentation for one- and two-stage architectures with different colors. (1) First row: ground-truth labels corresponding to each video sequence frame. (2) Second row: one stage with transformer TCN, which regards it as

one basic block and stacks six blocks. (3) Third row: one stage with DASPP, which regards it as one basic block and stacks six blocks. (4) Fourth row: our two-stage TSRN includes one prediction block and five refinement blocks

errors in predictions compared with simply stacking the same subnetworks.

### 4.3 Effectiveness of the multi-head self-attention mechanism

To demonstrate the effectiveness of the multi-head self-attention mechanism in the transformer TCN, we report the performance of our TSRN and its variants with and without the multi-head self-attention mechanism. As shown in Table 3, the multi-head self-attention mechanism effectively improves the quality of action segmentation results. Particularly, the improvement of the F1 score by up to 26% on the Breakfast dataset indicates that attaching the multi-head self-attention mechanism in the transformer encoder block can capture temporal relations to alleviate over-segmentation errors. In addition, the numbers of heads and layers in the transformer encoder block are listed in Table 4.

The multi-head self-attention mechanism helps understand the local information of action segments and infer

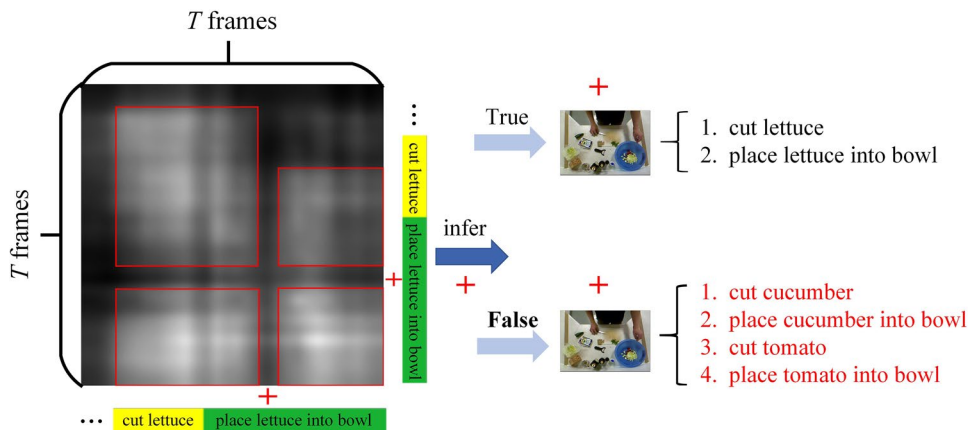
**Table 5** Effect of the number of dilated residual layers ( $K$ ) in the transformer TCN on the results for the 50Salads dataset

	F1@{10, 25, 50}			Edit	Acc
$K = 0$	72.6	71.1	64.8	66.4	81.4
$K = 4$	75.6	73.8	68.7	66.6	83.2
$K = 6$	75.7	73.4	64.2	68.1	81.9
$K = 8$	80.7	79.0	70.7	73.0	83.9
$K = 10$	81.3	79.3	72.3	74.1	84.3
$K = 11$	<b>84.9</b>	<b>83.5</b>	<b>77.3</b>	<b>79.3</b>	<b>84.5</b>
$K = 12$	82.6	80.3	75.0	75.2	84.1

The bold values indicate the highest value for each metric (column) in that table/comparison

actions around neighboring action segments in global semantic relations. For example, we select a video sequence from the 50Salads dataset and obtain the attention matrix from the standard deviation. The horizontal and vertical axes represent the frames of a video sequence.

**Fig. 7** Visualization of the attention matrix for the multi-head self-attention mechanism



**Table 6** Performance of the TSRN using DASPP and variants with and without channel attention (Channel), spatial pyramid pooling module (SPP), and spatial attention (Spatial)

Channel	SPP	Spatial	F1@{10, 25, 50}			Edit	Acc
×	×	×	74.5	73.2	63.1	70.2	80.0
√	√	×	77.7	75.1	66.6	70.5	80.4
×	√	√	80.1	77.1	69.4	72.1	82.7
√	×	√	80.6	78.5	70.1	73.8	82.2
√	√	√	<b>84.9</b>	<b>83.5</b>	<b>77.3</b>	<b>79.3</b>	<b>84.5</b>

The bold values indicate the highest value for each metric (column) in that table/comparison

The visualization results in Fig. 7 show that for a query frame “+,” the neighboring areas (red boxes) indicate that the multi-head self-attention mechanism focuses on the meaningful locations of adjacent action segments. That is, actions that are irrelevant to the local semantics (e.g., “cut cucumber” and “cut tomato” are incorrect predictions for the local information of processing lettuce) cannot be predicted in the consecutive video sequence. Hence, the multi-head self-attention mechanism effectively models the temporal relations in each action segment.

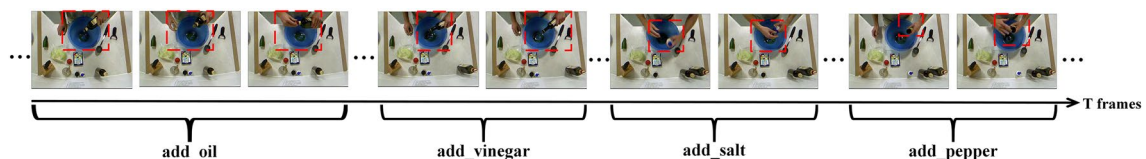
To better understand how the transformer TCN constantly receives global features after capturing the temporal relations among local continuous action segments by utilizing the multi-head self-attention mechanism, we present the performance of different residual layers after the transformer encoder block on the 50Salads dataset. As shown in Table 5, increasing  $K$  from 0 to 11 typically improves the performance, especially for F1 scores and segmental edit distance. This indicates expanding the receptive fields after the transformer encoder block achieves better results because the global features are obtained gradually, which demonstrates that capturing long-range dependencies in the transformer TCN plays an essential role in the first-stage prediction.

#### 4.4 Effectiveness of the DASPP

In this section, we validate the effectiveness of the DASPP in our TSRN, which captures multiscale features with large temporal receptive fields and precisely revises segmentation boundaries. As presented in Table 6, both the SPP and attention modules (CAM and SAM) greatly improve the action segmentation performance. Compared with the variant without any module in the DASPP, the variant with CAM and

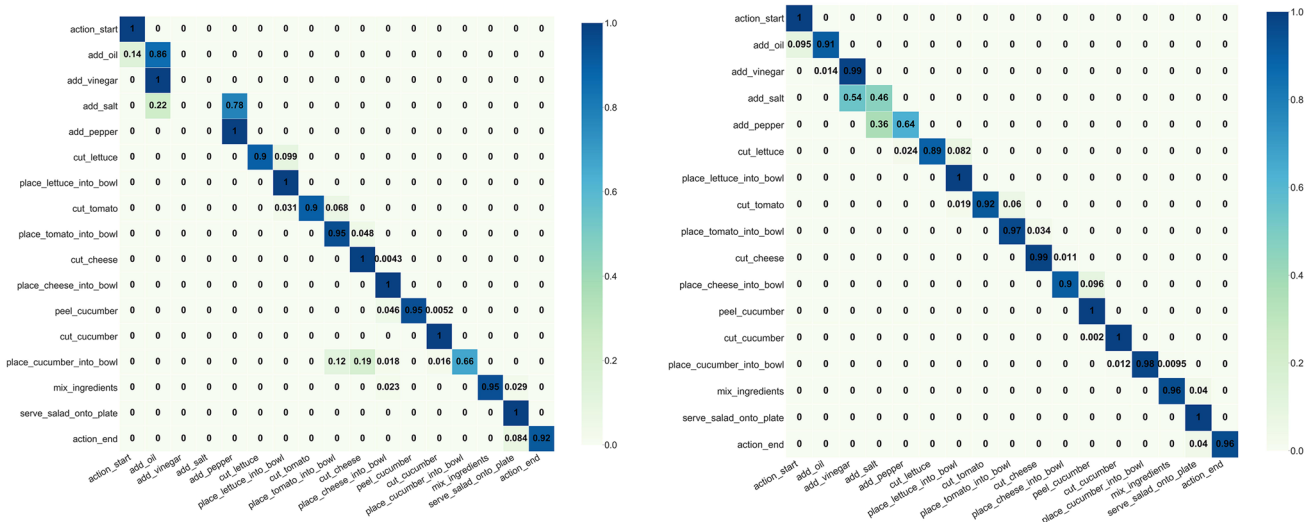
SAM brings a 7% improvement in F1@50, which indicates that channel attention and spatial attention are essential for focusing on the local features of action segments. In Fig. 8, although the action appearances are similar in each action segment, the CAM and SAM in the DASPP helps the information flow within the network by learning which information is worth emphasizing (valuable information in the image is indicated by a red box) or which is inhibiting.

Furthermore, to achieve the fusion of local and global features from the video sequence, we added three detection headers built on the top of the three feature maps after the CAM module in DASPP at different scales for fusing multiscale features in the input sequence. The results in Table 6 show that the SPP improves the F1 score from 70.1% to 77.3% and the segmental edit distance from 73.8% to 79.3%, demonstrating that fusing the multiscale features makes it easier to classify the ambiguous frames of action segments than using only two attention modules (CAM and SAM). In Fig. 9, we compare the baseline model with the TSRN to visualize the effectiveness of DASPP. MS-TCN [11] is based on dilated temporal convolutional networks that adopt a multi-stage architecture similar to that of our TSRN (i.e., iteratively refining the prediction from the backbone model several times to obtain the revised version segmental results), and it is the baseline model of the TSRN. In contrast to the TSRN, the MS-TCN [11], which does not contain DASPP, wrongly recognized “add\_vinegar” as “add\_oil,” “add\_salt” as “add\_oil” and “add\_pepper.” Moreover, “place\_cucumber\_into\_bowl” is misidentified as “place\_tomato\_into\_bowl” and “cut\_cheese”. This phenomenon shows that the DASPP produces more accurate action boundaries and identifies different adjacent action segments when segmenting indistinguishable appearance actions.



**Fig. 8** Illustration of how the CAM and SAM in the DASPP focus on the local features of action segments





**Fig. 9** Confusion matrix results for the test set of the 50Salads dataset. (Left) MS-TCN [11] baseline model without the DASPP. (Right) TSRN with the DASPP

**Table 7** Performance of different  $\alpha$  values in the focal loss  $L_{focal}$  on the 50Salads and GTEA datasets

50Salads	F1@{10, 25, 50}			Edit	Acc
$\alpha = 0.10$	81.2	79.7	73.2	75.4	83.3
$\alpha = 0.15$	<b>84.9</b>	<b>83.5</b>	<b>77.3</b>	<b>79.3</b>	<b>84.5</b>
$\alpha = 0.20$	84.3	81.2	74.4	76.1	82.5
GTEA					
$\alpha = 0.15$	86.4	83.9	76.5	79.4	78.8
$\alpha = 0.20$	88.5	86.4	78.8	81.3	79.2
$\alpha = 0.25$	<b>89.4</b>	<b>87.8</b>	<b>80.1</b>	<b>84.9</b>	<b>80.6</b>

The bold values indicate the highest value for each metric (column) in that table/comparison

**Table 8** Comparison of the combinations of loss functions on the 50Salads and Breakfast datasets

50Salads	F1@{10, 25, 50}			Edit	Acc
$L_{focal} + L_{self}$	79.5	77.4	68.2	71.7	83.1
$L_{cls} + L_{reg}$	78.4	75.6	66.6	71.0	80.9
$L_{cls} + L_{reg} + L_{focal}$	78.0	75.6	66.4	70.6	80.4
$L_{cls} + L_{reg} + L_{focal} + L_{self}$	<b>84.9</b>	<b>83.5</b>	<b>77.3</b>	<b>79.3</b>	<b>84.5</b>
Breakfast					
$L_{focal} + L_{self}$	30.3	24.5	14.3	32.3	36.5
$L_{cls} + L_{reg}$	59.5	53.8	43.0	63.1	69.8
$L_{cls} + L_{reg} + L_{focal}$	<b>75.4</b>	<b>70.3</b>	<b>56.2</b>	<b>75.0</b>	<b>71.6</b>
$L_{cls} + L_{reg} + L_{focal} + L_{self}$	58.4	53.3	42.5	61.5	71.2

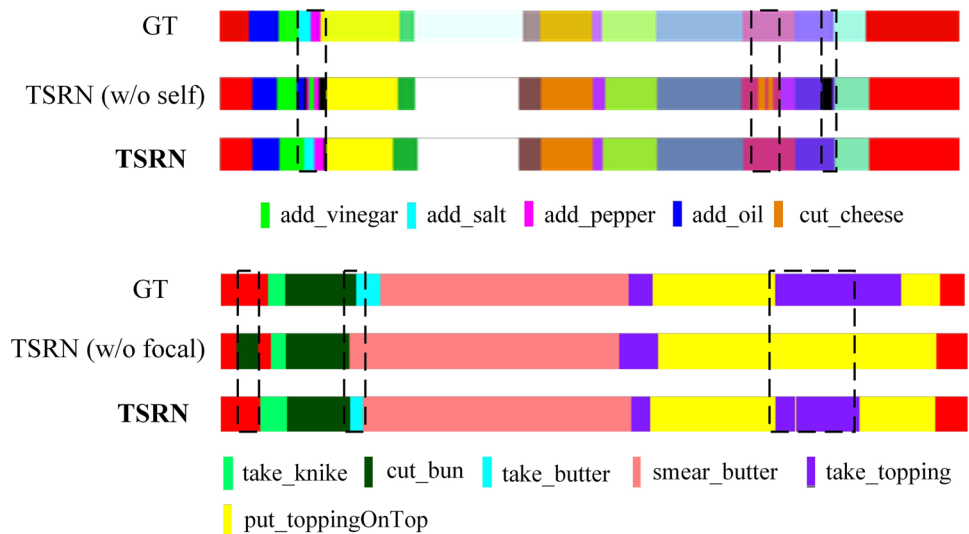
The bold values indicate the highest value for each metric (column) in that table/comparison

### 4.5 Effectiveness of the joint loss

We first make a parameter ablation study of the focal loss  $L_{focal}$  in the proposed joint loss function for the subsequent experiments. In the focal loss,  $\gamma$  is used to adjust the rate to smooth the hard-to-classify samples during training, which is fixed to 2 for all datasets because of the advanced performance.  $\alpha$  is a critical parameter (shown in Eq. (14)) to balance the distribution of well-classified and hard-to-classify samples. Table 7 shows the performance in the focal loss  $L_{focal}$  of different  $\alpha$  values on the 50Salads and GTEA datasets. From Table 7, we observe that the best weighting factor is  $\alpha = 0.15$  for the 50Salads dataset, which denotes the proportion of hard-to-classify samples estimated at 85%. This forces the model to focus on hard-to-classify frames during training, alleviating the ambiguity of identifying action boundaries. The sample distributions of different datasets are different, so we need to select suitable  $\alpha$  values for various datasets. Hence, we observe that when  $\alpha = 0.15$  and  $\alpha = 0.25$  for the Breakfast and GTEA datasets, respectively, we achieve excellent performance.

To verify the effectiveness of the joint loss function, we report the performance of TSRN and its variants with and without focal loss and auxiliary self-supervision signals while training the two-stage architecture on the 50Salads and Breakfast datasets. Table 8 compares the performance of each combination of loss functions. The proposed joint loss function improves the F1 score by up to 10.9% and the segmental edit distance by 8.7% on the 50Salads dataset after incorporating auxiliary self-supervised loss  $L_{self}$  and focal loss  $L_{focal}$ . After training with self-supervision signals on the 50Salads dataset, our TSRN outperforms the

**Fig. 10** Qualitative comparison of action segmentation results. **a** Comparison of TSRN with and without auxiliary self-supervision loss on the 50Salads dataset. **b** Comparison of TSRN trained with and without focal loss on the Breakfast dataset



same network without self-supervision signals by 4% in all evaluation metrics on the 50Salads dataset, which indicates the auxiliary self-supervision task is able to improve the segmental results and reduce over-segmentation errors. The performance of the model that contains auxiliary self-supervision loss trained on the Breakfast dataset does not work effectively because the correct temporal relations in the video sequence have been severely disrupted. Exchanging frames in the Breakfast dataset that include the maximum action classes may exacerbate the burden of classifying the exchanged frames and the original frames while ignoring how to correct the exchanged frames. Moreover, our joint loss leads to a remarkable improvement in the Breakfast dataset results after a focal loss has been added, i.e., there is nearly 11% improvement in all metrics, except for frame-wise accuracy. Note that the performance of the combination of the cross-entropy and truncated mean squares error losses is relatively bad because of the noises during training, while focal loss is effective in balancing the frequency of different action classes and smoothing the transition of action probabilities.

The qualitative comparison in Fig. 10 shows that the auxiliary self-supervision task is essential for boosting temporal relations and revising the labels of incorrectly labeled

action segments, reducing over-segmentation errors at the boundaries of action segments. Moreover, focal loss plays an indispensable role in balancing the frequency of each action class, which shows the potential for enhancing the generalizability of the model.

#### 4.6 Effectiveness of the number of refinement blocks

To illustrate the effectiveness of stacking several refinement blocks over the second stage in the TSRN, we compare the segmental results from the different refinement blocks. To declare that the improvement of our model is due to the design choice instead of simply raising the model's capacity, we compare the proposed TSRN with its variants by the following evaluation metrics: F1 score, segmental edit distance, frame-wise accuracy, floating-point operations per second (FLOPs), and parameters (Params) in Table 9.

Table 9 tabulates the performance of different numbers of refinement blocks on the 50Salads dataset. The results show that increasing the number of refinement blocks from 3 to 5 significantly improves the performance due to the expansion of receptive fields. However, the performance starts to diminish by adding the 6th refinement block,

**Table 9** Comparison of the numbers of refinement blocks on the 50Salads dataset. The floating-point operations per second (FLOPs) are one random video sequence with 4000 frames. Params and FLOPs are calculated on a single Tesla V100 GPU with batch size=1

	F1@{10, 25, 50}			Edit	Acc	Params (M)	FLOPs (G)
TSRN (3 refinement blocks)	80.9	79.0	71.7	73.7	85.0	0.95	4.88
TSRN (4 refinement blocks)	82.9	79.5	73.6	76.1	85.1	1.13	5.71
<b>TSRN (5 refinement blocks)</b>	<b>84.3</b>	<b>83.5</b>	<b>76.7</b>	<b>78.3</b>	<b>85.4</b>	1.31	6.27
TSRN (6 refinement blocks)	82.5	80.6	72.6	74.7	84.7	<b>1.49</b>	<b>6.96</b>

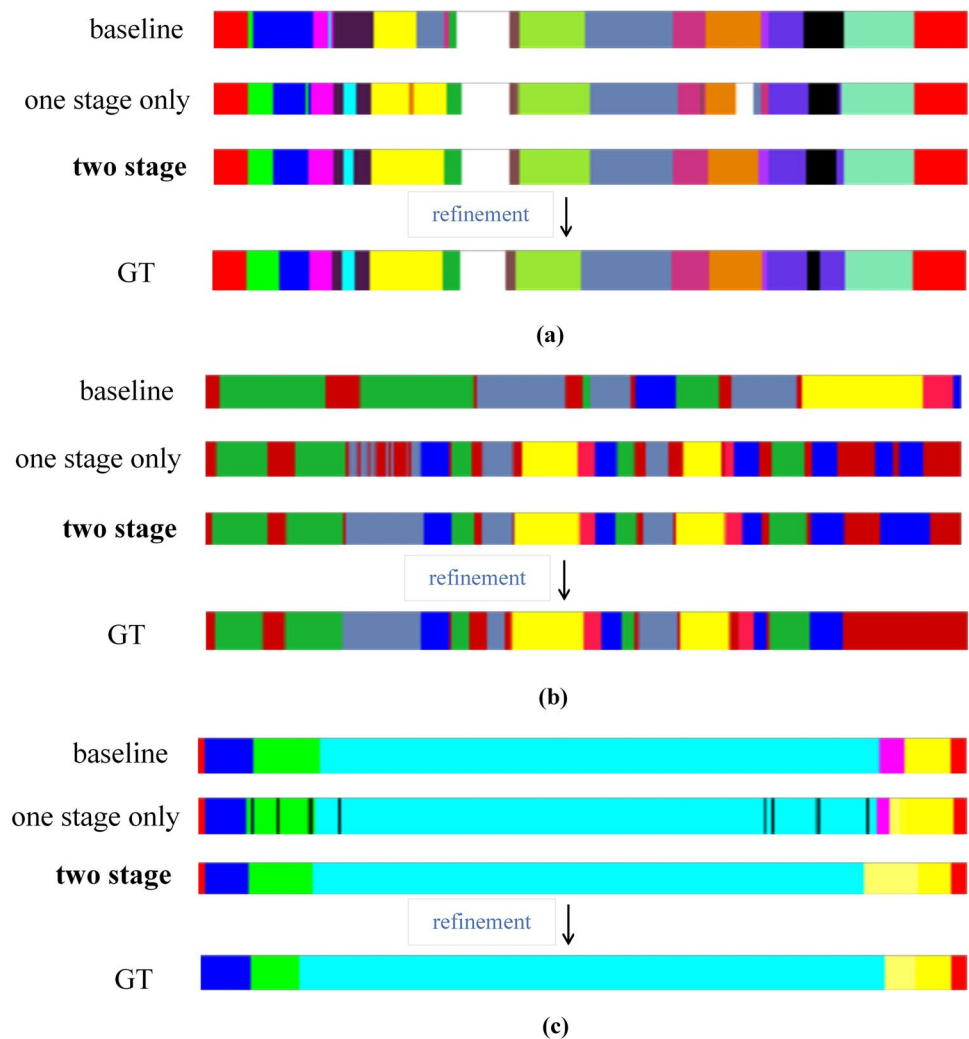
The bold values indicate the highest value for each metric (column) in that table/comparison

**Table 10** Performance of the TSRN and state-of-the-art methods on the 50Salads, GTEA, and Breakfast datasets

Methods	50Salads			GTEA			Breakfast		
	F1@{10, 25, 50}	Edit	Acc	F1@{10, 25, 50}	Edit	Acc	F1@{10, 25, 50}	Edit	Acc
	ST-CNN [48]	55.9	37.1	45.9	58.7	54.4	41.9	60.6	–
Bi-LSTM [9]	62.8	47.0	55.6	66.5	59.0	43.6	55.5	–	–
ED-TCN [10]	68.0	52.6	59.8	72.2	69.3	56.0	64.7	–	43.3
TriconNet [5]	70.1	56.6	62.8	72.2	76.0	71.1	64.8	–	–
TRN [21]	70.2	56.3	63.7	67.1	63.7	51.9	59.9	–	–
TDRN [21]	72.9	57.2	66.0	79.2	74.4	62.7	70.1	–	–
LCDC [50]	73.8	–	66.9	75.4	–	–	65.3	–	65.3
Huang et al. [24]	75.4	63.9	67.5	82.6	–	–	–	57.5	58.7
Zhang et al. [22]	78.4	66.7	71.0	86.7	84.3	72.7	82.3	–	–
RPGaussian [49]	78.3	66.4	72.3	88.3	86.8	74.6	78.5	62.0	63.5
MS-TCN [11]	76.3	64.5	67.9	86.8	83.4	69.8	76.3	52.6	66.3
MS-TCN + [14]	80.7	70.1	74.3	88.8	85.7	76.0	80.1	64.1	67.6
G2L [26]	80.3	69.8	73.4	89.9	87.3	75.8	78.5	76.3	70.8
DTGRM [16]	79.1	66.1	72.0	87.8	86.6	72.9	77.6	68.7	68.3
G-FRNet [23]	78.0	67.0	71.4	89.1	85.7	72.8	76.7	71.1	67.7
BCN [12]	82.3	74.0	74.3	88.5	87.1	77.3	79.8	68.7	70.4
<b>TSRN</b>	<b>83.4</b>	<b>76.7</b>	<b>78.3</b>	<b>89.4</b>	<b>87.8</b>	<b>80.1</b>	<b>80.6</b>	<b>75.4</b>	<b>71.6</b>

The bold values indicate the highest value for each metric (column) in that table/comparison

**Fig. 11** Qualitative results for the temporal action segmentation task from the representative samples on the **a** 50Salads, **b** GTEA, and **c** Breakfast datasets obtained from the baseline method without a two-stage refinement (MS-TCN [11]), one-stage architecture before the refinement process, proposed two-stage TSRN and GT (ground truth)



caused by over-fitting during the training process. Besides, as the number of blocks grows, the computational burden also becomes onerous, which is reflected in the FLOPs and Params evaluation metrics. To balance the model performance and the computational power, 5 refinement blocks are selected in TSRN for all experiments. Our TSRN has 1.31 million parameters and requires 6.27 GB FLOPs, and its computational burden is affordable considering the available hardware.

#### 4.7 Comparison with state-of-the-art methods and results

We compare our proposed TSRN with state-of-the-art methods on the 50Salads, GTEA, and Breakfast datasets. Table 10 shows that the proposed TSRN is exceptional to the state-of-the-art methods with a competitive F1 score and segmental edit distance, particularly with a large margin of up to 22.8% for the F1 score and 10.4% for the segmental

edit distance when compared with the baseline model [11]. Moreover, it is worth noting that our framework also exceeds the existing methods on the 50Salads dataset for all evaluation metrics. The F1 score and segmental edit distance are fundamental metrics that evaluate the accuracy of segmentation. The F1@10 performances of TSRN and the temporal convolutional encoder-decoder with bilinear pooling operation [22, 49] are similar on the GTEA dataset. However, TSRN increases by 7.4% in terms of F1@50 because our network is suitable for identifying actions that largely overlap with ground truth segments.

We crucially compare TSRN with the following seven methods that adopt a similar multi-stage architecture as TSRN: MS-TCN [11], MS-TCN++ [14], Huang, et al. [24], DTGRM [16], G2L [26], G-FRNet [23], and BCN [12]. MS-TCN++ [14] is an extended version of the baseline model MS-TCN [11], and it uses the same backbone model and parameters as [11]. The methods of DTGRM [16] and Huang et al. [24] are built on top of [11] and refine the original results using graph convolutional networks, which



are related to our TSRN as they model relations among action segments in a similar manner. G-FRNet [23] forces the refinement process to correct the errors in the previous segmental results. G2L [26] is a proposed global-to-local scheme that is akin to TSRN, which captures long- and short-term features in a hierarchical structure. Table 10 reveals that TSRN substantially outperforms MS-TCN [11] and MS-TCN + + [14] in significant improvements on all datasets, indicating the necessity of temporal reasoning in temporal convolutional networks. As for the Breakfast dataset, our TSRN outperforms DTGRM [16] and Huang et al. [24] with a large margin, i.e., 6%-17% increases in F1 score and segmental edit distance, which demonstrates that TSRN not only models the temporal relations but also refines ambiguous action boundaries in temporal action segmentation. Even though the performances of BCN [12] and G2L [26] are close to that of TSRN, the notable improvement in segmental edit distance reveals that TSRN penalizes over-segmentation errors, whereas these two methods still have a large room for improvement.

It should be mentioned that the above seven methods based on dilated TCNs have impressive research value for action segmentation. However, TSRN further overcomes the limitations by gradually developing the global and local features in a two-stage strategy. Although the accuracy of TSRN is more competitive than that of the state-of-art models, the results of our TSRN on the GTEA and Breakfast datasets are potentially not optimal. This is because the number of action instances in each video in GTEA is more than those in other datasets, and the number of videos in Breakfast is the maximum of all datasets, resulting in the arduous task of predicting fine-grained actions with strong reasoning abilities. Our TSRN is based on the architecture of dilated TCNs that makes the improvements of frame-wise temporal reasoning, while it does not consider over-segmentation errors and the ambiguous boundary problem in instance-wise action predictions in long videos. i.e., current works have not paid attention to increasing the accuracy of predicted action instances. In future research, we will investigate more effective methods to model instance-wise temporal relations and enhance the training process with an augmentation strategy to promote the robustness of our model.

The qualitative results for the representative examples of temporal action segmentation are shown in Fig. 11. Predictions without two-stage architecture or one stage before the refinement process appear as ambiguous frames in action boundaries. Over-segmentation errors occurred in the predictions due to lacking semantic connections, resulting in incorrect short intervals in a continuous video sequence. In contrast, the TSRN mitigates these problems through its two-stage architecture (Fig. 11a, b, and c). Compared with MS-TCN [11] and one-stage architecture before the refinement process, the action boundary of the

video sequence predicted by our two-stage TSRN is more precise and closer to the ground-truth labels. At the same time, for actions unrelated to global semantics, TSRN can identify and revise them into the correct labels (e.g., the color-coding in the black of Fig. 11c indicates the actions are unrelated to the entire video).

Despite the significant progress of the TSRN, some actions may be confusing because of the incredibly high similar motion appearance of the images. As can be seen in Fig. 9 Right, it is difficult to distinguish “add\_salt” and “add\_pepper” in the process of making salads, which leads us to pay more attention to identifying similar actions in a video.

## 5 Conclusion

We present the TSRN for the temporal action segmentation task, consisting of two stages: transformer TCN for focusing on the semantic association among the action segments and gradually receiving global features and DASPP for fusing multiscale features to alleviate over-segmentation errors. In addition, we introduce a joint loss that further refines the predictions. The TSRN outperforms state-of-the-art methods on three challenging datasets. The results imply that the performance of the TSRN achieves better than simply stacking more convolutional networks. To perfect our work, we will find a practical method to classify similar actions in a continuous video sequence. We hope this work could promote the development of action understanding and provide the mentality for potential applications, such as action parsing and action reasoning.

**Acknowledgements** This work was supported in part by the National Natural Science Foundation of China (Grant Number: 51935005), Basic Scientific Research Project (Grant Number: JCKY20200603C010), Natural Science Foundation of Heilongjiang Province of China (Grant Number: LH2021F023), and Science & Technology Planned Project of Heilongjiang Province of China (Grant Number: GA21C031)

## Declarations

**Competing Interests** The authors declare that they have no known competing financial interests or personal relationships that have influenced the work reported in this manuscript.

**Data availability** All data included in this study are available upon request by contact with the corresponding author.

## References

1. Febin IP, Jayasree K, Joy PT (2020) Violence detection in videos for an intelligent surveillance system using MoBSIFT and movement filtering algorithm. *Pattern Anal Appl* 23(2):611–623

2. Pan Z, Liu S, Sangaiah AK, Muhammad K (2018) Visual attention feature (VAF): a novel strategy for visual tracking based on cloud platform in intelligent surveillance systems. *J Parallel Distr Com* 120:182–194
3. Stenum J, Rossi C, Roemmich RT (2021) Two-dimensional video-based analysis of human gait using pose estimation. *Plos Comput Biol* 17(4):e1008935
4. Feichtenhofer C, Pinz A, Wildes RP (2017) Spatiotemporal multiplier networks for video action recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, pp 4768–4777
5. Ding L, Xu C (2017) Tricorner: A hybrid temporal convolutional and recurrent network for video action segmentation. arXiv preprint [arXiv:1705.07818](https://arxiv.org/abs/1705.07818)
6. Lin J, Gan C, Han S (2019) Tsm: Temporal shift module for efficient video understanding. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), IEEE, pp 7083–7093
7. Simonyan K, Zisserman A (2014) Two-stream convolutional networks for action recognition in videos. arXiv preprint [arXiv:1406.2199](https://arxiv.org/abs/1406.2199)
8. Tran D, Bourdev L, Fergus R, Torresani L, Paluri M (2015) Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the IEEE international conference on computer vision (ICCV), IEEE, pp 4489–4497
9. Singh B, Marks TK, Jones M, Tuzel O, Shao M (2016) A multi-stream bi-directional recurrent neural network for fine-grained action detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, pp 1961–1970
10. Lea C, Flynn MD, Vidal R, Reiter A, Hager GD (2017) Temporal convolutional networks for action segmentation and detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, pp 156–165
11. Farha YA, Gall J (2019) Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, pp 3575–3584
12. Wang Z, Gao Z, Wang L, Li Z, Wu G (2020) Boundary-aware cascade networks for temporal action segmentation. In: Proceedings of the European conference on computer vision (ECCV), Springer, pp 34–51
13. Ishikawa Y, Kasai S, Aoki Y, Kataoka H (2021) Alleviating over-segmentation errors by detecting action boundaries. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision (WACV), IEEE, pp 2322–2331
14. Li SJ, Abufarha Y, Liu Y, Cheng MM, Gall J (2020) Ms-tcn++: Multi-stage temporal convolutional network for action segmentation. *IEEE Trans Pattern Anal.* <https://doi.org/10.1109/TPAMI.2020.3021756>
15. Chen MH, Li B, Bao Y, Alregib G, Kira Z (2020) Action segmentation with joint self-supervised temporal domain adaptation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, pp 9454–9463
16. Wang D, Hu D, Li X, Dou D (2021) Temporal Relational Modeling with Self-Supervision for Action Segmentation. In: Proceedings of the aaai conference on artificial intelligence (AAAI). 35(4), pp 2729–2737
17. Stein S, Mckenna SJ (2013) Combining embedded accelerometers with computer vision for recognizing food preparation activities. In: Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing, pp 729–738
18. Fathi A, Ren X, Rehg JM (2011) Learning to recognize objects in egocentric activities. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, pp 3281–3288
19. Kuehne H, Arslan A, Serre T (2014) The language of actions: Recovering the syntax and semantics of goal-directed human activities. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, pp 780–787
20. Oord AVD, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K (2016) Wavenet: A generative model for raw audio. arXiv preprint [arXiv:1609.03499](https://arxiv.org/abs/1609.03499).
21. Lei P, Todorovic S (2018) Temporal deformable residual networks for action segmentation in videos. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition (CVPR), IEEE, pp 6742–6751
22. Zhang Y, Tang S, Muandet K, Jarvers C, Neumann H (2019) Local temporal bilinear pooling for fine-grained action parsing. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, pp 12005–12015
23. Wang D, Yuan Y, Wang Q (2020) Gated forward refinement network for action segmentation. *Neurocomputing* 407:63–71
24. Huang Y, Sugano Y, Sato Y (2020) Improving action segmentation via graph-based temporal reasoning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, pp 14024–14034
25. Chen MH, Li B, Bao Y, Alregib G (2020) Action segmentation with mixed temporal domain adaptation. In: Proceedings of the IEEE/CVF Winter conference on applications of computer vision (WACV), IEEE, pp 605–614
26. Gao SH, Han Q, Li ZY, Peng P, Wang L, Cheng MM (2021) Global2local: Efficient structure search for video action segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, pp 16805–16814
27. Kitaev N, Cao S, Klein D (2018) Multilingual constituency parsing with self-attention and pre-training. arXiv preprint [arXiv:1812.11760](https://arxiv.org/abs/1812.11760).
28. Cheng X, Qiu G, Jiang Y, Zhu Z (2021) An improved small object detection method based on Yolo V3. *Pattern Anal Appl* 24(3):1347–1355
29. Kuehne H, Gall J, Serre T (2016) An end-to-end generative framework for video segmentation and recognition. In: Processing of the IEEE/CVF Winter conference on applications of computer vision (WACV), IEEE, pp 1–8
30. Arnab A, Dehghani M, Heigold G, Sun C, Lucic M, Schmid C (2021) Vivit: A video vision transformer. In: Proceedings of the IEEE/CVF International conference on computer Vision (ICCV), IEEE, pp 6836–6846
31. Zheng S, Lu J, Zhao H, Zhu X, Luo Z, Wang Y, Fu Y, Feng J, Xiang T, Torr PHS (2021) Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition (CVPR), IEEE, pp 6881–6890
32. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. *Adv Neural Inf Process Syst* 30:5998–6008
33. He L, Wen S, Wang L, Li F (2021) Vehicle theft recognition from surveillance video based on spatiotemporal attention. *Appl Intell* 51(4):2128–2143
34. Wang J, Xiong H, Wang H, Nian X (2020) ADSCNet: asymmetric depthwise separable convolution for semantic segmentation in real-time. *Appl Intell* 50(4):1045–1056
35. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 7132–7141
36. Woo S, Park J, Lee JY, Kweon IS (2018) Cbam: Convolutional block attention module. In: Proceedings of the European conference on computer vision (ECCV), Springer, pp 3–19

37. Lin TY, Dollar P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 2117–2125
38. Lin TY, Goyal P, Girshick R, He K, Dollar P (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision (ICCV), IEEE, pp 2980–2988
39. He K, Zhang X, Ren S, Sun J (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans Pattern Anal Mach Intell* 37(9):1904–1916
40. Tang K, Li FF, Koller D (2012) Learning latent temporal structure for complex event detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 1250–1257
41. Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady* 10(8):707–710
42. Carreira J, Zisserman A (2017) Quo vadis, action recognition? a new model and the kinetics dataset. In: Proceedings of the IEEE Conference on computer vision and pattern recognition (CVPR), IEEE, pp 6299–6308
43. Donahue J, Anne Hendricks L, Guadarrama S et al (2015) Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 2625–2634
44. Vinyals O, Toshev A, Bengio S et al (2015) Show and tell: A neural image caption generator. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 3156–3164
45. Tao L, Zappella L, Hager GD et al (2013) Surgical gesture segmentation and recognition. In: 2013 International conference on medical image computing and computer-assisted intervention (MICCAI), Springer, pp 339–346
46. Rohrbach M, Amin S, Andriluka M et al (2012) A database for fine grained activity detection of cooking activities. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 1194–1201
47. Cheng Y, Fan Q, Pankanti S et al (2014) Temporal sequence modeling for video event detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 2227–2234
48. Lea C, Reiter A, Vidal R, et al (2016) Segmental spatiotemporal cnns for fine-grained action segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV), Springer, pp 36–52
49. Zhang Y, Muandet K, Ma Q (2019) Frontal low-rank random tensors for fine-grained action segmentation. arXiv preprint [arXiv:1906.01004](https://arxiv.org/abs/1906.01004).
50. Mac KNC, Joshi D, Yeh RA, Xiong J, Feris RS, Do MN (2019) Learning motion in feature space: locally-consistent deformable convolution networks for fine-grained action detection. In: Proceedings of the IEEE/CVF International conference on computer vision (ICCV), IEEE, pp 6282–6291
51. Richard A, Kuehne H, Gall J (2017) Weakly supervised action learning with rnn based fine-to-coarse modeling. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 754–763
52. Li Z, Sun Y, Zhang L et al (2021) CTNet: context-based tandem network for semantic segmentation. *IEEE Trans Pattern Anal Mach Intell* 44(12):9904–9917
53. Zhou H, Li Z, Ning C, et al (2017) Cad: Scale invariant framework for real-time object detection. In: Proceedings of the IEEE international conference on computer vision workshops, pp 760–768

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.