ASFormer: Transformer for Action Segmentation

Fangqiu Yi
chinayi@pku.edu.cn
Hongyu Wen
hermerawen@pku.edu.cn
Tingting Jiang
ttjiang@pku.edu.cn

NELVT
Department of Computer Science
Peking University, China

Abstract

Algorithms for the action segmentation task typically use temporal models to predict what action is occurring at each frame for a minute-long daily activity. Recent studies have shown the potential of Transformer in modeling the relations among elements in sequential data. However, there are several major concerns when directly applying the Transformer to the action segmentation task, such as the lack of inductive biases with small training sets, the deficit in processing long input sequence, and the limitation of the decoder architecture to utilize temporal relations among multiple action segments to refine the initial predictions. To address these concerns, we design an efficient Transformer-based model for the action segmentation task, named ASFormer, with three distinctive characteristics: (i) We explicitly bring in the local connectivity inductive priors because of the high locality of features. It constrains the hypothesis space within a reliable scope, and is beneficial for the action segmentation task to learn a proper target function with small training sets. (ii) We apply a pre-defined hierarchical representation pattern that efficiently handles long input sequences. (iii) We carefully design the decoder to refine the initial predictions from the encoder. Extensive experiments on three public datasets demonstrate the effectiveness of our methods. Code is available at https://github.com/ChinaYi/ASFormer.

1 Introduction

Algorithms for automatic detection and segmentation of human activities are crucial for applications such as home security, healthcare, and robot automatic. Different from the action classification task that tries to classify a short trimmed video into a single action label, the goal of the action segmentation task is to assign an action label for each frame for a minutes-long untrimmed video. Instead of using raw RGB video sequences as the input, action segmentation methods operate on pre-extracted frame-wise feature sequences and focus on modeling the temporal relations among frames. Transformers, originally designed for the machine translation task [41], have achieved great performance for almost all natural language processing(NLP) tasks over the past years. Very recently, many researchers also show the potential of pure or hybrid Transformer models for many vision tasks, including image classification [4, 10, 40, 45, 50], action classification [1], segmentation [44, 46, 52], *et al.*

The task of action segmentation is similar to NLP tasks, since both of them are sequence-to-sequence prediction tasks. With the success of Transformer-based models in modeling the relations among elements in sequential data, one would expect the Transformer-based models to be highly effective for the action segmentation task as well.

However, there are three major concerns when solving the action segmentation task by the vanilla Transformer [41] :

1. **Due to the small size of training sets, the lack of inductive biases** of the vanilla Transformer becomes the bottleneck of applying it to the action segmentation problem. The lack of inductive biases broadens the family of functions they can represent [6]. However, it requires a large amount of training data. Compared to the NLP task and other vision tasks, the training set of the action segmentation task is relatively small, making it difficult to learn a target function from a large hypothesis space.

2. **Due to the deficit of self-attention for the long input video, the Transformer is hard to form an effective representation**. At initialization, the self-attention layer cast nearly uniform attention weights to all the elements in the sequence. However, the input video for the action segmentation task usually lasts for thousands of frames, much longer than the image-patch sequences in other vision tasks. Due to the length of videos, it is challenging for the self-attention layer to learn appropriate weights that focus on meaningful locations [53]. The deficit of each self-attention layer further raises a serious issue: it is hard for those self-attention layers in one Transformer model to cooperate with each other to form an effective representation for the input.

3. **The original encoder-decoder architecture of the Transformer does not meet the refinement demand of the action segmentation task.** Temporal relations among multiple action segments play an important role in the action segmentation task, *e.g.* the action after *take bottle* and *pour water* usually to be *drink water*. Given an initial prediction, previous works usually apply additional TCNs [11, 27] or GCNs [15, 43] over the initial prediction to perform a refinement process to boost the performance. However, the decoder in the vanilla encoder-decoder architecture is not designed for such usage.

In this paper, we will address the above three concerns in our proposed ASFormer, as shown in Fig. 1. For the first concern, we observe that one property of the action segmentation task is the high locality of features because every action occupies continued timestamps. Thus, the **local connectivity inductive bias** is important to the action segmentation task. It constrains the hypothesis space within a reliable scope, and is beneficial to learn a proper target function with small training sets. We bring in such strong inductive priors by applying additional temporal convolutions in each layer. For the second concern that the Transformer with serials of self-attention layers is hard to form an effective representation over the long input sequence, we constrain each self-attention layer with a **pre-defined hierarchical representation pattern**, which forces the low-level self-attention layers to focus on the local relations at first and then gradually enlarges their footprints to capture longer dependencies in high-level layers. The local-to-global process assigns specific responsibilities to each self-attention layer, so that they can cooperate better to achieve faster convergence speed and higher performance. Such a hierarchical representation pattern also reduces the total space and time complexity to make our model scalable. Finally, we propose **a new design of the decoder** to acquire the refined predictions. The cross-attention mechanism in the decoder

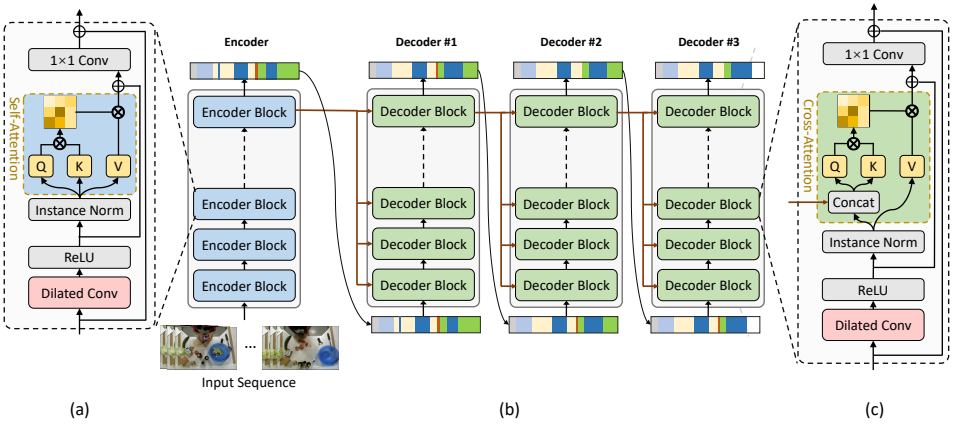


Figure 1: (b). An overview of the ASFormer model which consists of an encoder and several decoders to perform an iterative refinement. For the encoder, it receives video sequences and outputs initial predictions. Encoder consists of serials of encoder blocks with pre-defined hierarchical representation patterns. For the decoder, it receives predictions as the input and has similar architecture with encoder. (a). Each encoder block consists of a feed-forward layer (dilated temporal convolution) and a self-attention layer with residual connections. (c). The decoder block uses cross-attention mechanism to bring in information from the encoder.

allows every position in the encoder to attend over all positions in the refinement process, simultaneously avoiding the disturbance of the encoder to the learned feature space in the refinement phase.

Experiments are conducted on three common public datasets, including the 50Salads [37], Breakfast [19] and GTEA [12]. The experimental results demonstrate the proposed solution is capable of dealing with small training datasets and long videos with thousands of frames. The design of decoders also takes advantage of temporal relations among multiple action segments to help get smoother and accurate predictions. To summarize, the main contributions of this work include: 1) An exploration for Transformer on action segmentation task with three distinctive characteristics: the explicitly introduced local connectivity inductive bias, pre-defined hierarchical representation pattern, and new design of the decoder; 2) state-of-the-art action segmentation results on three public datasets.

2 Related Work

Action Segmentation. Earlier approaches detect action segments using a sliding window and filter out redundant hypotheses with non-maximum suppression [18, 33]. Other approaches model the temporal action sequence with Conditional Random Fields [30, 34, 39], Markov model [20, 38] and RNN [9, 29, 35, 42, 49] to classify framewise actions. In recent years, motivated by the success of temporal convolution in speech synthesis, most of the works try to apply various temporal convolution networks for the action segmentation task, such as the encoder-decoder temporal convolution [8, 23], deformable temporal convolution [24], or dilated temporal convolution [11, 16]. Other works build on top of these TCNs. Wang *et al*. [47] reuse the structure of [11] as cascade stages. Huang *et al*. [15] and Wang *et al*. [43] propose a graph-based temporal reasoning module, which can be added to

the top of action segmentation models. Chen *et al*. [5] propose to exploit unlabeled auxiliary videos by shaping this problem as a domain adaptation (DA) problem. Very recently, Ishikawa1*et al*. [17] alleviate over-segmentation errors by detecting action boundaries. Singhania*et al*. [36] follow a coarse-to-fine structure with an implicit ensemble of multiple temporal resolutions. However, a single convolutional layer does not connect all pairs of input and output positions, which remains room for improvement.

Transformer. Transformer [41] is originally designed for natural language processing tasks, which relies on the self-attention mechanism to build dependencies among elements. Due to the strong capability of capturing global information, hybrid Transformer models have also been successfully applied to a variety of vision problems, including image classification [9, 10, 40, 45, 50], action classification [1], segmentation [44, 46, 52], *et al*. More recently, some researches study the efficient version of Transformer models, which explore attention restrictions to local windows, such as Swin [28], BigBird [51]. In this paper, we explore the application of Transformer based models on the action segmentation task.

Action Detection. Different from action segmentation task, the action detection task aims at localizing the start/end of the action and recognizing it in an untrimmed video. The One-stage methods [25, 26] draw on the SSD [27] method in object detection and design end-to-end action detection networks with the similar feature pyramid structures. Two-stage methods [3, 48] adopt the Faster-RCNN [31] architecture, including proposal generation and proposal classification.

3 Methods

In this work, we propose ASFormer to tackle the action segmentation task, as shown in Fig. 1. Our ASFormer adapts an encoder-decoder structured Transformer. Given the pre-extracted frame-wise video feature sequence, the encoder will first predict the initial action probability for each frame. Then the initial predictions will be passed to multiple successive decoders to perform an incremental refinement. In Sec. 3.1, we first illustrate the structure of **encoder**, showing how we deal with small training datasets and long videos with thousands of frames. After that, we introduce the design of **decoders** and our way to take advantage of temporal relations among multiple action segments for refinement in Sec. 3.2. Finally, in Sec. 3.3, we introduce the implementation and training details of our ASFormer.

3.1 Encoder

The input for the encoder is the pre-extracted feature sequences of size $T \times D$, where T is the video length and D is the feature dimension. The first layer of the encoder is a fully connected layer that adjusts the dimension of the input feature. Then, this layer is followed by serials of encoder blocks. After that, a fully connected layer will output predictions $y_e \in R^{T \times C}$ from the last encoder block, where C denotes the number of action classes.

Each encoder block contains two sub-layers. The first is a feed-forward layer, and the second is a single-head self-attention layer. We employ a residual connection around each of the two sub-layers, followed by instance normalization and ReLU activation, as shown in Fig. 1(a). Different from the vanilla Transformer, we use a dilated temporal convolution as the feed-forward layer instead of a point-wise fully connected layer. This design is inspired by the properties of the action segmentation task, which are a) lack of the large training dataset for completely free inductive bias. b) high locality of features since every action

occupies continued timestamps in the input video. In contrast to the fully connected layer, the temporal convolution layer can bring beneficial local inductive bias to our model.

The self-attention layer is hard to be learned to focus on meaningful locations over thousands of frames. For an input video, those self-attention layers are difficult to cooperate with each other to form an effective representation. To mitigate this issue, we pre-define a hierarchical representation pattern. Such a hierarchical pattern is inspired by modern neural network design: one would first focus on the local feature and then gradually enlarge the receptive field to capture the global information. For example, CNNs achieve such patterns with serials of pooling layers to enlarge the receptive field in higher layers; or use dilated convolutions with a gradually increasing dilation rate. Motivated by the success of such a hierarchical pattern, we constrain the receptive fields of each self-attention layer within a local window with size w (*e.g.* for a frame t, we only calculate the attention weights with the frames that within its local window). The size of the local window is then doubled at each layer i (i.e., $w = 2^i, i = 1, 2...$). Meanwhile, we also double the dilation rate of the temporal convolution layer with the encoder depth increasing, keeping consistent with the self-attention layer.

For an encoder with J blocks, the whole approximate memory usage for a vanilla Transformer is $O(J \cdot T \cdot T)$, where T is the video length. With the hierarchical representation pattern, we reduce the total space complexity to $O((2-\varepsilon) \cdot 2^J \cdot T)$, where ε is a small number. In our settings, we use $J = 9$, where $2^J = 512$ is almost 10 times smaller than T. Compared to the vanilla Transformer, our ASFormer is applicable to receiving long input sequences.

3.2 Decoders

Temporal relations among multiple action segments play an important role in the action segmentation task. There are some prior relationship between actions segments, *e.g.* the action after *take bottle* and *pour water* usually to be *drink water*. As previous works showed, applying additional TCNs [11, 47] or GCNs [15, 43] over the initial prediction to perform a refinement process can boost the performance. In this section, we illustrate how the newly designed decoder carries out the refinement task for the initial predictions by the encoder in one-forward-pass. For a better explanation, we first introduce a single decoder, and naturally extend it to a multiple version to perform iterative refinement.

A Single Decoder. The input for the decoder is the initial output by the encoder. The first layer of the decoder is a fully connected layer to adjust the dimension, and then followed by a serial of decoder blocks. The architecture of each decoder block is shown in Fig. 1(c). Similar to the encoder, we use temporal convolution as the feed-forward layer and the hierarchical pattern is also applied for the cross-attention layer.

Compared to the self-attention layer, the cross-attention has the following difference: The query Q and key K are obtained from the concatenation of the output from the encoder and previous layer, while the value V is only obtained from the output of the previous layer. The cross-attention mechanism allows every position in the encoder to attend overall positions in the refinement process by generating the attention weights. The feature space V is completely transformed from the input predictions, and will not be disturbed with the participant of the encoder, because the generated attention weights are only used to perform a linear combination within V. Such design is inspired by the previous work [11], where they show the refinement process is very sensitive to the disturbance of the learned feature space from the predictions.

Multiple Decoders. One would naturally expand the single decoder to a multiple version to perform iterative refinement. In the multiple version, the input of each decoder is from the previous one, as shown in Fig. 1(b).

The cross-attention mechanism allows to bring in external information to guide the refinement process. We hope to gradually reduce the weight of external information to avoid the problem of error accumulation. For the input x in each decoder block, we use a weighted residual connection for the output of the feed-forward layer and the cross-attention layer:

$$\begin{aligned} out &= feed\_forward(x) \\ out &= \alpha * cross\_att(out) + out \end{aligned} \tag{1}$$

We set $\alpha = 1$ for the first decoder and then exponentially decrease α for rest decoders.

3.3 Loss Function & Implementation details

The loss function is a combination of classification loss $\mathcal{L}_{cls}$ for each frame and smooth loss $\mathcal{L}_{smo}$ [11]. The classification loss is a cross-entropy loss, while the smooth loss calculates the mean squared error over the frame-wise probabilities. The final loss function $\mathcal{L}$ is,

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda \mathcal{L}_{smo} = \frac{1}{T}\sum_{t} -log(y_{t,\hat{c}}) + \lambda \frac{1}{TC}\sum_{t}\sum_{c}(y_{t-1,c} - y_{t,c})^2$$

where $y_{t,\hat{c}}$ is the the predicted probability for the ground truth label $\hat{c}$ at time t. λ is a balance weight that is set to 0.25 in our experiments. Finally to train the complete model, the sum of the losses over the encoder and all decoders is minimized.

The final ASFormer consists of one encoder and three decoders, while each encoder and the decoder contains nine blocks. The dimension of the first fully connected layer in the encoder and decoder is set to 64, as well as the feature dimension in each encoder and decoder block. Moreover, a special dropout layer is applied to the input feature of the encoder, which randomly drops the entire feature channel with a dropout rate of 0.3. In all experiments, we train the model for 120 epochs through Adam optimizer with a learning rate 0.0005.

4 Dataset

50Salads [37] dataset contains 50 videos with 17 action classes of users making a salad and has been used for both action segmentation and detection. On average, each video contains 20 action instances and is about 6.4 minutes long. These activities were performed by 25 actors where each actor prepared two different salads. For evaluation, we perform 5-fold cross-validation and report the average results.

GTEA [12] dataset contains 28 videos of 11 action classes of daily activities in a kitchen, like take or pour, performed by four subjects. On average, each video has 20 action instances and is about half-minute long on average. We use the standard four different train-test splits as previous works by leaving one subject out. Similar to 50Salads, the average results of four splits are reported.

Breakfast [19] dataset is the largest and the most challenging dataset among the three datasets with 1712 videos. The videos were recorded in 18 different kitchens, showing breakfast preparation-related activities. Overall, there are 48 different actions, and each

video contains six action instances on average. We use the standard 4-fold cross-validation for evaluation and report the average results.

For all the three datasets, we use the I3D [2] model, which is trained on kinetics [2] dataset, to pre-extract feature sequences as in previous works [7, 11, 15, 23, 43, 47]. The dimension of the I3D feature for each frame is 2048-d. The following three evaluation metrics are used to evaluate the performance: frame-wise accuracy(Acc.), segmental edit score (Edit), and segmental overlap F1 score with threshold k/100, denoted as F1@k.

5 Experiments

5.1 Impact of position encoding and multi-head self-attention

In terms of implementation details, our ASFormer has two differences compared to the vanilla Transformer. First, the position encoding is not applied for the input of both encoder and decoders. Second, vanilla Transformer usually adapts multi-head attention that concatenates the output from multiple single-head attention to enhance the feature transformation ability. Instead, we only use one single-head attention in each encoder/decoder block. We conduct the following two experiments to demonstrate the introduction of temporal convolution makes our model free from position encoding and multi-head self-attention.

The first experiment studies whether position encoding is still needed for ASFormer. The results on 50Salads dataset are shown in Table 1. For the temporal convolution, the position encoding is even counterproductive. When the position encoding is only applied for the encoder, the performance already drops a lot. Further, when the position encoding is applied for both encoder and decoders, the performance is even worse. The possible reason is that the temporal convolution already has the ability to model the relative positional relationships. The redundant absolute position encoding might be harmful to the temporal convolutions to learn the feature embedding. Thus, we drop the position encoding from our model. The second experiment studies the impact of multi-head self-attention. Table 2 shows the comparison of multi-head self-attention with the different numbers of heads on 50Salads. For each head in the multi-head attention, the feature dimension is the same as the single-head attention. We can observe that the single head self-attention achieves comparable performance with the multiple ones. This is because the convolution operation has a similar form with the multi-head self-attention operation (*e.g.* a 64-filters convolution operator also concatenates the output from 64 1-filter convolutions). Considering the extra computation and memory budgets brought by the multi-head self-attention, we use the single-head self-attention by default.

Table 1: Comparison of whether using position encoding on 50Salads.

	F1@{10, 25, 50}			Edit	Acc.
w/o pe (ours)	**85.1**	**83.4**	**76.0**	**79.6**	**85.6**
+ pe encoder	80.2	78.0	70.0	73.0	83.6
+ pe decoder	78.1	76.7	69.4	72.0	84.1

Table 2: Comparison of multi-head self-attention with different number of heads on 50Salads.

	F1@{10, 25, 50}			Edit	Acc.
single head (ours)	85.1	**83.4**	76.0	**79.6**	85.6
two heads	85.0	83.0	75.9	78.2	**85.9**
three heads	**85.2**	83.2	**76.8**	78.1	85.3

5.2 Effect of the local connective inductive bias

In this section, we study the effect of bringing in local connective inductive bias with the temporal convolution. For comparison, we use an MLP as the feed-forward layer as the same

as the vanilla Transformer. It is worth noting that when removing the temporal convolution from our model, the position encoding is needed to bring in position information as the vanilla transformer does. In order to directly compare the two solutions, we only use the encoder part for the experiments to avoid the impact of the refinement process. Results on the 50Salads dataset are shown in Table 3. We can observe that when MLP is used as the feed-forward layer, the performance drops greatly, especially on F1 scores and Edit score, which denotes that the model fails to model the temporal relationship among frames to produce smoother and consistent predictions. This also demonstrates that the fusion of neighboring local information plays an important role in the performance.

Table 3: Comparison of MLP with position encoding and temporal convolution as the feed-forward layer on 50Salads. Results on the the **encoder** are reported.

	F1@{10, 25, 50}			Edit	Acc.
MLP+PE	27.6	25.3	19.9	20.0	74.2
Conv (ours)	**53.1**	**51.4**	**47.0**	**43.3**	**85.7**

5.3 Effect of the hierarchical representation pattern

The hierarchical representation pattern plays an important role in our ASFormer. To demonstrate its effectiveness, we conduct a non-hierarchical version by setting the size of the local window in all attention layers to 512, which is the largest window size in the last encoder/decoder block (out of memory if we directly set the size of the local window in all attention layers to the video length). The non-hierarchical version allows each self-attention layer to span their attention weights 'freely'. Experiments on the 50Salads dataset are shown in Table 4. The performance of the non-hierarchical version drops significantly.

To better understand why there is such a huge performance gap, we show some visualization results in Fig. 2. For an anchor query frame, we plot its attention weights in each self-attention layer in the encoder (The attention weights are normalized with min-max[1].). Fig. 2(a) shows the non-hierarchical version, while Fig. 2(b) shows the case that with the pre-defined hierarchical pattern. We have two observations. First, for the high-level blocks (*e.g.* block #9) that have the same window size, there is much more activation (blue ribbon) in the non-hierarchical version after the min-max normalization. This means that the attention weights in many locations have close values and are more similar to a trivial uniform distribution. In contrast, self-attention trained with the hierarchical pattern tends to focus on several meaningful locations. Secondly, the 'freely' attention cannot automatically learn a hierarchical pattern from data. As shown in Fig. 2(a), the low-level blocks do not cast attention on their neighbors when the self-attention layer is not constrained.

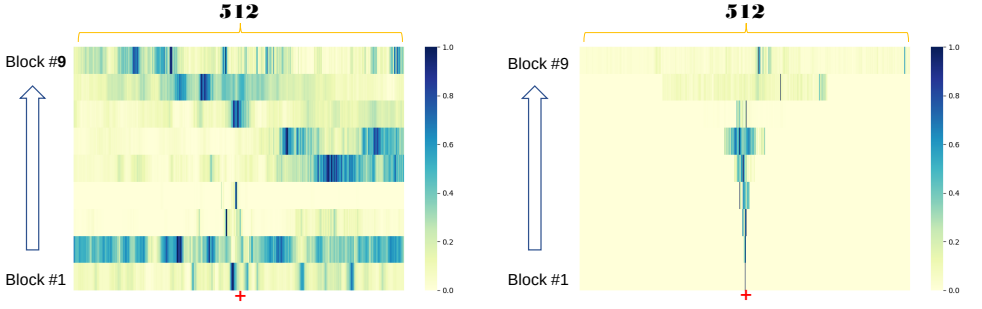
5.4 Effect of the multiple decoders

To demonstrate that our decoder takes advantage of temporal relations among multiple action segments for refinement, we conduct an ablation study about stacking the different numbers of decoders. Results on the 50Salads dataset are shown in Table 5. The decoders largely boost the performance compared to the encoder, and we achieve the best results when stack-

[1] $x = (x - min)/(max - min)$

Table 4: Comparison of hierarchical representation pattern and the non-hierarchical pattern (by setting the size of local window in all attention layers to 512) on 50Salads.

	F1@{10, 25, 50}			Edit	Acc.
non-hierarchical	64.2	61.5	55.1	59.5	76.8
hierarchical (ours)	**85.1**	**83.4**	**76.0**	**79.6**	**85.6**



(a) non-hierarchical (b) hierarchical

Figure 2: The visualization of attention weights for an anchor frame (red +) in each encoder block, more visualization can be found in the supplementary material. (a) The non-hierarchical (by setting the window size to 512 in all blocks). (b) With the hierarchical pattern.

ing three decoders. To show the iterative refinement process, we further plot the predictions in the encoder and all decoders, as shown in the supplementary material.

5.5 Ablations of the number of blocks

The number of blocks J in the encoder/decoder is an important hyper-parameter, where more blocks bring larger receptive fields but also lead to higher memory costs as introduced in Sec. 3.1. We conduct an ablation study about the different number of blocks in the encoder/decoder on the 50Salads dataset and report the peak value of the GPU memory cost for batch size 1 during training. Results are shown in Table 6. We achieve higher performance with the increasing number of blocks until $J = 9$. Using more than 9 blocks ($J = 10$) slightly improves the frame-wise accuracy but does not increases the F1 scores. Meanwhile, the GPU memory cost increase dramatically with the large J. Thus, we choose $J = 9$ for all our experiments.

Table 5: Comparison of stacking different number of decoders on 50Salads.

	F1@{10, 25, 50}			Edit	Acc.
Encoder Only	53.1	51.4	47.0	43.3	85.7
One Decoder	79.5	77.4 ,	71.6	71.5	**86.8**
Two Decoders	83.9	82.8	76.8	76.7	**86.8**
Three Decoders (ours)	**85.1**	**83.4**	**76.0**	**79.6**	85.6
Four Decoders	84.0	82.2	75.8	76.5	84.3

Table 6: Comparison of different number of blocks J on 50Salads.

J	F1@{10, 25, 50}			Edit	Acc.	GPU Mem.
7	82.9	81.5	74.0	76.0	84.5	∼2.1G
8	84.4	82.8	75.4	78.2	85.4	∼2.5G
9	**85.1**	83.4	76.0	**79.6**	85.6	∼3.5G
10	84.7	**83.6**	**76.5**	79.5	**86.4**	∼6.1G

5.6 Comparison with SOTA

This section compares the proposed ASFormer to the state-of-the-art methods on three datasets: 50Salads, GTEA, and Breakfast datasets. The results are presented in Table 7. Our model

achieves the state-of-the-art methods on the three datasets compared to previous work. The results on three metrics highlight the ability of our ASFormer to obtain accurate and smooth predictions.

Table 7: Comparison with the state-of-the-art on 50Salads, GTEA and the Breakfast datasets. **Bold** and underlined denote the highest and the second value in each column.

	50Salads					GTEA					Breakfast				
	F1@{10,25,50}			Edit	Acc.	F1@{10,25,50}			Edit	Acc.	F1@{10,25,50}			Edit	Acc.
IDT+LM [32]	44.4	38.9	27.8	45.8	48.7	-			-	-	-			-	-
ST-CNN [22]	55.9	49.6	37.1	45.9	59.4	58.7	54.4	41.9	-	60.6	-			-	-
Bi-LSTM [33]	62.6	58.3	47.0	55.6	55.7	66.5	59.0	43.6	-	55.5				-	-
ED-TCN [23]	68.0	63.9	52.6	59.8	64.7	72.2	69.3	56.0	-	64.0	-			-	43.3
HTK [21]	-			-	-	-			-	-	-			-	50.7
TCFPN [8]	-			-	-	-			-	-	-			-	52.0
SA-TCN [2]	-			-	-	-			-	-	-			-	50.0
HTK(64) [20]	-			-	-	-			-	-	-			-	56.3
TDRN [24]	72.9	68.5	57.2	66.0	68.1	79.2	74.4	62.7	74.1	70.1	-			-	-
SSA-GAN [13]	74.9	71.7	67.0	69.8	73.3	80.6	79.1	74.2	76.0	74.4	-			-	-
MS-TCN [11]	76.3	74.0	64.5	67.9	80.7	85.8	83.4	69.8	79.0	76.3	52.6	48.1	37.9	61.7	66.3
DTGRM [15]	79.1	75.9	66.1	72.0	80.0	87.8	86.6	72.9	83.0	77.6	68.7	61.9	46.6	68.9	68.3
BCN [10]	82.3	81.3	74.0	74.3	84.4	88.5	87.1	77.3	84.4	**79.8**	68.7	65.5	55.0	66.2	70.4
Gao *et al.* [14]	80.3	78.0	69.8	73.4	82.2	89.9	87.3	75.8	84.6	78.5	74.9	69.0	55.2	73.3	70.7
ASRF [17]	84.9	**83.5**	**77.3**	79.3	84.5	89.4	87.8	**79.8**	83.7	77.3	74.3	68.9	56.1	72.4	67.6
ASFormer	**85.1**	83.4	76.0	**79.6**	**85.6**	**90.1**	**88.8**	79.2	**84.6**	79.7	**76.0**	**70.6**	**57.4**	**75.0**	**73.5**

Table 8: Comparison of ASRF(build upon *MS-TCN*) and ASRF*(build upon our *ASFormer*) on 50Salads dataset.

	F1@{10, 25, 50}			Edit	Acc.
ASRF$_{(MS\text{-}TCN)}$	84.9	83.5	77.3	79.3	84.5
ASRF*$_{(ASFormer)}$	**86.8**	**85.4**	**79.3**	**81.9**	**85.9**

Table 9: Comparison of ASFormer and MS-TCN on number of parameters, FLOPs and GPU memory cost on 50Salads dataset.

	#params (M)	FLOPs(G)	GPU Mem.
MS-TCN	0.799	4.79	∼1.7G
ASFormer	1.134	6.80	∼3.5G

It is worth noting that our ASFormer can serve as a strong backbone model and thus can be easily adapted by other works. To demonstrate that, we replace the original TCN-based backbone model MS-TCN [11] in ASRF [17] with our ASFormer. The new model, denoted as ASRF*, achieves even higher results on the 50Salads dataset than the original ASRF [17], as shown in Table 8. Besides the performance, we further compare with the popular TCN-based backbone MS-TCN [11] in terms of the computation cost, as shown in Table 9. The additional computational burden of our ASFormer is acceptable with the modern chips. Thus, our ASFormer provides an alternative backbone choice for the community to build new models.

6 Conclusion

In this paper, we explore the application of Transformer-based models on the action segmentation task. We propose three major concerns and respective solutions in our ASFormer. The superior results on three public datasets demonstrate the effectiveness of our method, and provide a new solution for the community to address the action segmentation task.

Acknowledgement. This work was partially supported by the Natural Science Foundation of China under contracts 62088102. We also acknowledge the High-Performance Computing Platform of Peking University for providing computational resources.

References

[1] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *CoRR*, abs/2102.05095, 2021.

[2] João Carreira and Andrew Zisserman. Quo Vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.

[3] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A. Ross, Jia Deng, and Rahul Sukthankar. Rethinking the Faster R-CNN architecture for temporal action localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1130–1139, 2018.

[4] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. CrossViT: Cross-attention multi-scale vision transformer for image classification. *CoRR*, abs/2103.14899, 2021.

[5] Min-Hung Chen, Baopu Li, Yingze Bao, and Ghassan AlRegib. Action segmentation with mixed temporal domain adaptation. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 594–603, 2020.

[6] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations (ICLR)*, 2020.

[7] Rui Dai, Luca Minciullo, Lorenzo Garattoni, Gianpiero Francesca, and François Bremond. Self-attention temporal convolutional network for long-term daily living activity detection. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–7, 2019.

[8] Li Ding, Chenliang Xu, and Juergen Gall. Weakly-supervised action segmentation with iterative soft boundary assignment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 6508–6516, 2018.

[9] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(4):677–691, 2017.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL https://arxiv.org/abs/2010.11929.

[11] Yazan Abu Farha and Jurgen Gall. MS-TCN: multi-stage temporal convolutional network for action segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3575–3584, 2019.

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

[13] Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Fine-grained action segmentation using the semi-supervised action GAN. *Pattern Recognition*, 98:107039, 2020.

[14] Shang-Hua Gao, Qi Han, Zhong-Yu Li, Pai Peng, Liang Wang, and Ming-Ming Cheng. Global2Local: Efficient structure search for video action segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[15] Yifei Huang, Yusuke Sugano, and Yoichi Sato. Improving action segmentation via graph-based temporal reasoning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14021–14031, 2020.

[16] Noureldien Hussein, Efstratios Gavves, and Arnold W.M. Smeulders. Timeception for complex action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 254–263, 2019.

[17] Yuchi Ishikawa, Seito Kasai, Yoshimitsu Aoki, and Hirokatsu Kataoka. Alleviating over-segmentation errors by detecting action boundaries. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[18] Svebor Karaman, Lorenzo Seidenari, and Alberto Del Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. In *European Conference on Computer Vision (ECCV), THUMOS Workshop*, 2014.

[19] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 780–787, 2014.

[20] Hilde Kuehne, Juergen Gall, and Thomas Serre. An End-to-End generative framework for video segmentation and recognition. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8, 2016.

[21] Hilde Kuehne, Alexander Richard, and Juergen Gall. Weakly supervised learning of actions from transcripts. *Computer Vision and Image Understanding*, 163:78–89, 2017.

[22] Colin Lea, Austin Reiter, René Vidal, and Gregory D. Hager. Segmental spatiotemporal CNNs for fine-grained action segmentation. In *European Conference on Computer Vision (ECCV)*, page 36–52, 2016.

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

[24] Peng Lei and Sinisa Todorovic. Temporal deformable residual networks for action segmentation in videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6742–6751, 2018.

[25] Xin Li, Tianwei Lin, Xiao Liu, Wangmeng Zuo, Chao Li, Xiang Long, Dongliang He, Fu Li, Shilei Wen, and Chuang Gan. Deep concept-wise temporal convolutional networks for action localization. In *Proceedings of the ACM International Conference on Multimedia (MM)*, page 4004–4012, 2020.

[26] Tianwei Lin, Xu Zhao, and Zheng Shou. Single shot temporal action detection. In *Proceedings of the ACM International Conference on Multimedia (MM)*, page 988–996, 2017.

[27] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

[28] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical vision transformer using shifted windows. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[29] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, 2015.

[30] Hamed Pirsiavash and Deva Ramanan. Parsing videos of actions with segmental grammars. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 612–619, 2014.

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39:1137–1149, 2017.

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

[33] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1194–1201, 2012.

[34] Qinfeng Shi, Li Cheng, Li Wang, and Alex Smola. Human action segmentation and recognition using discriminative semi-Markov models. *International Journal of Computer Vision (IJCV)*, 93:22–32, 2011.

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

[36] Dipika Singhania, Rahul Rahaman, and Angela Yao. Coarse to fine multi-resolution temporal convolutional network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[37] Sebastian Stein and Stephen J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of ACM International Joint Conference on Pervasive and Ubiquitous Computing*, page 729–738, 2013.

[38] Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1250–1257, 2012.

[39] Lingling Tao, Luca Zappella, Gregory Hager, and René Vidal. Surgical gesture segmentation and recognition. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 16, pages 339–46, 09 2013.

[40] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, pages 10347–10357, 2021.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, page 6000–6010, 2017.

[42] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, 2015.

[43] Dong Wang, Di Hu, Xingjian Li, and Dejing Dou. Temporal relational modeling with self-supervision for action segmentation. In *AAAI Conference on Artificial Intelligence*, 2021.

[44] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. MaX-DeepLab: End-to-End panoptic segmentation with mask transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[45] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *CoRR*, abs/2102.12122, 2021.

[46] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-End video instance segmentation with transformers. *CoRR*, abs/2011.14503, 2020.

[47] Zhenzhi Wang, Ziteng Gao, Limin Wang, Zhifeng Li, and Gangshan Wu. Boundary-aware cascade networks for temporal action segmentation. In *European Conference on Computer Vision (ECCV)*, 2020.

[48] Huijuan Xu, Abir Das, and Kate Saenko. R-C3D: Region convolutional 3D network for temporal activity detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 5794–5803, 2017.

[49] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision (IJCV)*, 126(2):375–389, 2018.

[50] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis E. H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-Token ViT: Training vision transformers from scratch on imagenet. *CoRR*, abs/2101.11986, 2021.

[51] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big Bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33, 2020.

[52] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[53] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for End-to-End object detection. *CoRR*, abs/2010.04159, 2020.