How Much Temporal Long-Term Context is Needed for Action Segmentation?

Emad Bahrami[1] Gianpiero Francesca[2] Juergen Gall[1,3]
[1]University of Bonn, Germany [2]Toyota Motor Europe, Belgium
[3]Lamarr Institute for Machine Learning and Artificial Intelligence, Germany

Abstract

Modeling long-term context in videos is crucial for many fine-grained tasks including temporal action segmentation. An interesting question that is still open is how much long-term temporal context is needed for optimal performance. While transformers can model the long-term context of a video, this becomes computationally prohibitive for long videos. Recent works on temporal action segmentation thus combine temporal convolutional networks with self-attentions that are computed only for a local temporal window. While these approaches show good results, their performance is limited by their inability to capture the full context of a video. In this work, we try to answer how much long-term temporal context is required for temporal action segmentation by introducing a transformer-based model that leverages sparse attention to capture the full context of a video. We compare our model with the current state of the art on three datasets for temporal action segmentation, namely 50Salads, Breakfast, and Assembly101. Our experiments show that modeling the full context of a video is necessary to obtain the best performance for temporal action segmentation.

1. Introduction

Temporal action segmentation can be used in many real-world applications such as monitoring production lines or studying animal behavior. In these settings, the videos can be very long and it is required to recognize the start and end of all actions that occur in a video as illustrated in Fig. 1.

Recently, combinations of temporal convolutional networks [1, 27] with self- and cross-attention from transformers [61, 4] have shown impressive results for temporal action segmentation. These works are in line with other hybrid models [15, 9, 61, 58] that combine the attention modules with convolutions to compensate for the lack of strong inductive bias of pure transformers. However, the emergence of datasets like Assembly101 [41], where subjects perform assembly tasks, poses a new challenge in the area of temporal action segmentation due to the existence of long videos

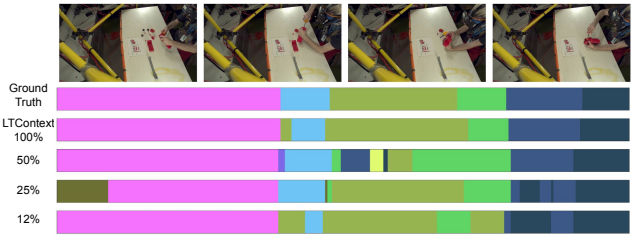

Figure 1: Datasets like Assembly101 contain long videos of assembly tasks and an action label needs to be predicted for each frame. The first row shows some frames of a video. The second row shows the ground-truth labels for all frames of the video where different colors correspond to different action labels. Rows 3-6 show the predictions of the proposed model for different amounts of long-term context where 100% means the temporal context of the full video.

that can last up to 25 minutes. Since modeling the long-term context of such a video is very expensive, Yi *et al*. [61] proposed to compute the attention for a local temporal window. In order to understand the impact of the window size on the temporal action segmentation accuracy, we analyze the impact of the window size on two datasets with long videos in Section 4.1. Fig. 1 shows some qualitative results of this study on Assembly101. Indeed, the results show that modeling the long-term context of an entire video is very important for temporal action segmentation.

Based on this finding, we revisit how temporal attention is modeled in transformer architectures for temporal action segmentation. [61, 4] use a hierarchy of temporal windows, which makes the training on long video sequences as they occur in Assembly101 [41] very expensive. Inspired by works that decompose attention over the spatial and temporal domain for short video clips [6, 62], we propose to iterate between computing windowed local attention and sparse long-term context attention such that both short and long-term context are modeled. This approach is particularly suitable for temporal action segmentation since the local attentions focus on the similarity or dissimilarity of features within an action segment or between neighboring action

segments whereas the long-term context attention focuses on the relations between actions within the entire video. The source code is available at `https://github.com/LTContext/LTContext`.

2. Related Work

The traditional sliding window with non-maxima suppression has been among the early approaches for action segmentation [19, 40]. [22, 20, 46] adopted hidden Markov Models (HMMs) for temporal modeling. [39] incorporates length and temporal context and uses dynamic programming for inference. Temporal convolutional networks (TCN) with temporal pooling are used by [23] to classify each video frame. Later, [1] introduced a multi-stage TCN capable of maintaining high temporal resolution, which is necessary for fine-grained recognition. [18] reduces the over-segmentation error by adding an action boundary regression network to refine the frame-wise results. [17] proposed another refinement method based on a graph convolutional network.

Transformers [52] were originally used for natural language processing and only rely on attention for sequence modeling. Recently, transformers have been widely adopted in vision [11, 57, 31, 54], speech recognition [15, 37], and action recognition [6, 2]. The original transformers suffered from two issues. First, the cost of the self-attention operation is quadratic with respect to the sequence length. This has been addressed by many methods that improve the memory efficiency of transformers [5, 34, 10]. We refer to [47] for a comprehensive survey of efficient transformers. For instance, restricting the attention to a fixed window size [5] is one approach. Multi-axis self-attention [51] combines block-attention with grid-attention, which is based on a spatial grid overlaid on the entire 2D space. Adapting sparse attention for capturing global information has shown to be an effective solution in vision tasks such as high-resolution image generation [63, 50], object detection [51, 60], and instance segmentation [51, 60]. MViT [12] uses a pooling operation to reduce the space-time resolution before attention and MViTv2 [28] improves MViT by adding the relative position along (x,y,t) and residual connections. The second issue of transformers is the poor generalization due to a relatively weak inductive bias [9, 49] compared to convolutional neural networks (CNN). Hybrid models that combine self-attention and convolution layers have thus been proposed for vision tasks [58, 9, 51, 57] and speech recognition [15, 37].

In action recognition, works such as [6, 62] adopt the idea of using attention for video understanding. In TimesFormer [6], the self-attention is first applied along the temporal dimension for each single patch (time attention), i.e., over all frames of a short video clip of a few seconds. In a second step, the attention is applied over the patches of each frame (space attention). VIDTR [62] also decomposes self-attention into spatial and temporal attention, but additionally down-samples the temporal dimension. In our work, we do not apply attention spatially and temporally, but we address the question of how temporal attention can be computed for very long sequences that last 25 minutes as it is required for temporal action segmentation.

For temporal action segmentation, [61] proposed an architecture, called ASFormer, which is based on a multi-stage TCN [1] and equips the temporal convolutions with a local window attention [5]. This is done in a hierarchy where the size of the local window grows with each layer. Recently, [4] proposed a decoder on top of the encoder that generates action segments in an autoregressive manner. It uses two heads where the head on top of the encoder predicts frame-wise probabilities and the head of the decoder predicts the sequence of actions. Finally, an alignment decoder fuses the output of the two heads and aligns the predicted sequence of actions to the frames. [3] introduced a hybrid Temporal Convolution Transformer (TCTr) where they adapt an action boundary detector to adaptively estimate attention from local neighboring frames. [59] proposed to use additional constraints during training, but the approach assumes that action sequences can be modelled by a directed acyclic graph, which does not allow that actions occur more than once in a video. Recently, multi-modal approaches that combine language models with vision transformers have been proposed. For instance, [26] uses prompt engineering to extract features from pre-trained vision-language models such as ActionCLIP [55].

3. Long-Term Context for Action Segmentation

Recently, transformers combined with temporal convolutional neural networks [61, 4, 3] have shown a very good performance for temporal action segmentation. For this task, the frame-wise labels $c_1, \ldots, c_T$, where $c_t \in \mathcal{C}$ and $C{=}|\mathcal{C}|$ denotes the number of action classes, need to be predicted for a given a video $X = (x_1, \ldots, x_T)$ with T frames, where x_t represents a feature map of size D at frame t. Since T can be very large, [61, 4, 3] limit the self-attention to a local temporal window.

In order to understand how much temporal long-term context is needed for temporal action segmentation, we limited the temporal input window and evaluated the quality of the temporal action segmentation in Section 4.1. The results in Fig. 4 show that temporal long-term context has a strong impact on the performance. Based on our analysis, we thus revisit the windowed attention of previous works for temporal action segmentation and propose to model the temporal long-term context of a video using sparse attentions [5, 38, 36]. Additionally, we equip our method with windowed attention to capture the locality between neighboring frames. In this way, we obtain a flexible design that is capable of providing long-term and local temporal context. While we describe first the Long-Term Context (LTCon-

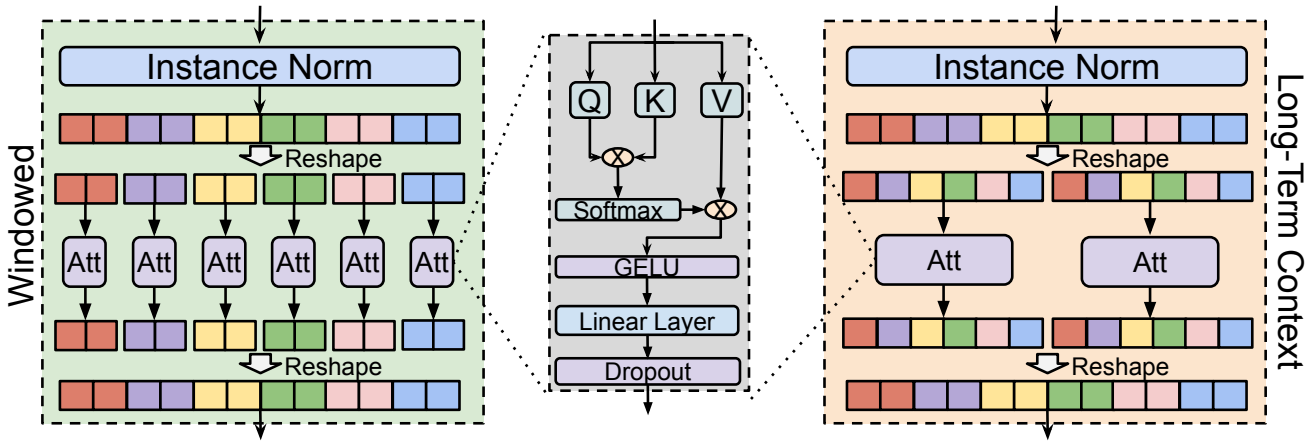

Figure 2: Illustration of windowed and long-term context attentions with a window of size 2. For the windowed attention, the sequence is partitioned into small windows and the attentions are computed for each window. For long-term context attention, the sequence is reordered such that the attentions are computed over the whole, but sparsely sampled sequence. After the attention, the output is reordered again to preserve the original order. Best viewed in color.

text) block in Section 3.1, the entire network is described in Section 3.2.

3.1. Temporal Context Attention

The self-attention blocks of transformers are advantageous over convolutions in the aggregation of global information. However, applying attention to long sequences such as untrimmed videos is impractical due to the quadratic complexity of the self-attention blocks. To address this issue, we adopt an attention mechanism where we leverage sparse and windowed attention to model long-term and local temporal context.

The attention function transforms the input into a query, key, and value and computes the output as a weighted sum of the values. For example, given a sequence of features, $X \in \mathbb{R}^{T \times D}$, the attention can be written as follows:

$$Attention(Q, K, V) = \text{softmax}_{row}\left(\frac{QK^T}{\sqrt{D}}\right)V \quad (1)$$

where $Q, K, V \in \mathbb{R}^{T \times D}$ are linearly transformed from X. Since T is very large for long sequences, we need to modify Q, K, V to enable the modeling of long-term and local temporal context as illustrated in Fig. 2.

Temporal Windowed Attention For the temporal Windowed Attention (WA), we partition the sequence into non-overlapping windows of size W. Fig. 2 illustrates the case for $W = 2$, but we use $W = 64$ in practice. The impact of W is evaluated in Section 4. Instead of computing the attention over the entire sequence of length T, we compute the attention $\frac{T}{W}$ times, where each query $Q \in \mathbb{R}^{W \times D}$ corresponds to each window. For the keys and values, we use an overlap where we use the next window in addition, *i.e.*, for each query Q we have $K, V \in \mathbb{R}^{2W \times D}$. We perform masking when K and V exceed the input sequence. We evaluate the impact of the overlap in Section 4.

Temporal Long-Term Context Attention For the temporal Long-Term Context (LTC) attention, the input is also partitioned into non-overlapping windows of size G. However, instead of computing the attention over each window, the attention is computed over all windows where from each window only one element is taken. In the illustration of Fig. 2 with $G = 2$, we compute the attention over the first feature of all windows and the attention over the second feature of all windows. In general, we compute the attentions for G queries $Q \in \mathbb{R}^{\frac{T}{G} \times D}$ where the keys and values are the same, *i.e.*, $K, V \in \mathbb{R}^{\frac{T}{G} \times D}$. The parameter G provides the flexibility to adjust the sparseness based on the available memory budget, *e.g.*, $G = 1$ corresponds to the case where the attention is applied over the full sequence. In practice, we use $G = 64$ and we evaluate the impact of G in Section 4.

LTContext block The top of Fig. 3 illustrates the entire LTContext block. As in previous works [1, 61, 4], we use a 1D dilated temporal convolution with kernel size 3, where the dilation factor increases by factor 2 for each layer. The dilated temporal convolution is followed by a Gaussian Error Linear Unit (GELU). In the LTContext block, we first use the windowed and then the long-term context attention, which are shown in Fig. 2. We evaluate the impact of the order in Section 4. Finally, we use a linear layer with a residual connection to output the features for each frame, $F \in \mathbb{R}^{T \times D}$.

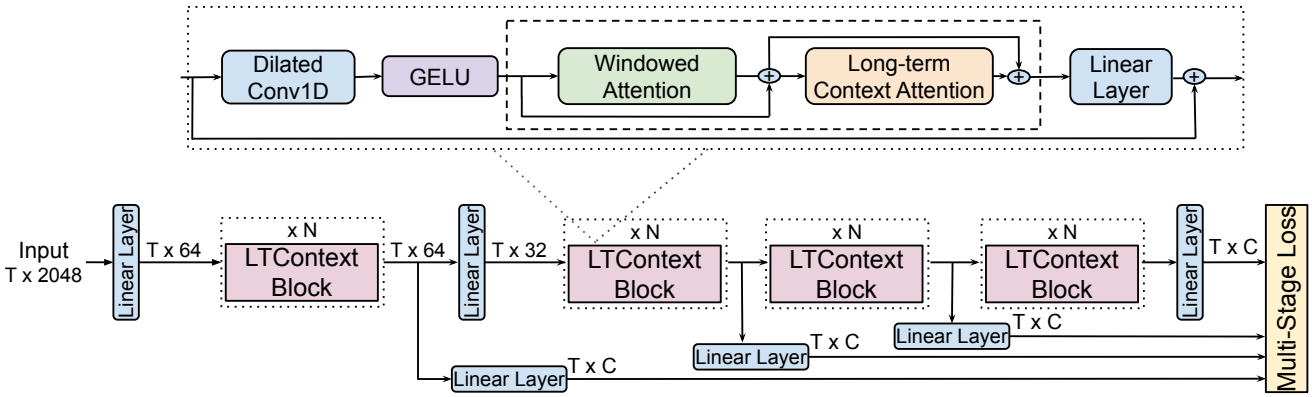

Figure 3: The network architecture of LTContext with LTContext blocks (top).

3.2. LTContext Architecture

The entire LTContext network is depicted in Fig. 3. For a fair comparison, we will use the features that are provided for the corresponding datasets as input. In all cases, the dimensionality of the features is 2048. As in previous works, we use a linear layer to reduce the dimensionality of the features to 64. The output of each LTContext block is the feature map $F \in \mathbb{R}^{T \times D}$. We repeat each LTContext block N times where the dilation factor of the temporal convolution increases in each layer. In practice, we use $N = 9$ and we evaluate the impact of N in Section 4. After the first N layers of LTContext blocks, we use an additional linear layer to reduce the dimensionality D further to 32. The dimensionality reduction reduces the number of parameters from 1.42 million to 0.72 million without reducing the accuracy. We also use an additional linear layer followed by a softmax layer to generate the frame-wise class probabilities $P \in \mathbb{R}^{T \times C}$.

We continue with three additional stages where each stage consists of N layers of LTContext blocks. Note that we reset the dilation factor to 1 for the temporal convolution at the beginning of each stage and we compute the frame-wise class probabilities $P \in \mathbb{R}^{T \times C}$ after each stage, which contributes to the multi-stage loss. We use the cross-entropy loss combined with the mean squared error smoothing loss as introduced by [1] and used in [27, 61] for a fair comparison. Inspired by [61], we use the cross-attention for the LTContext blocks in stages 2 to 4. Instead of using the features F for the queries and keys for windowed and long-term context attention, the predictions P are used. We thus have $Q \in \mathbb{R}^{W \times C}$, $K \in \mathbb{R}^{2W \times C}$, and $V \in \mathbb{R}^{2W \times D}$ for the windowed attention and $Q \in \mathbb{R}^{\frac{T}{G} \times C}$, $K \in \mathbb{R}^{\frac{T}{G} \times C}$, and $V \in \mathbb{R}^{\frac{T}{G} \times D}$ for the long-term context attention. While the cross-attention is not shown in Fig. 3, it only means that P is an additional input for the windowed and long-term context attention in stages 2-4. We evaluate the impact of the number of stages in Section 4.

4. Experiments

Datasets. We evaluate the performance of our proposed model on three challenging action segmentation datasets: 50Salads [45], Breakfast [21], and Assembly101 [41].

50Salads [45] contains 50 videos annotated with 17 action classes. On average, each video is 6.4 minutes long and has 18 action segments. Following previous works [1, 61, 4], we use five-fold cross-validation and report the average.

Breakfast contains 1,712 videos of breakfast preparation activities with an average length of 2.3 minutes. There are 48 action classes and each video has on average 6.6 action segments. For evaluation, we report the average of the 4 splits for cross-validation as in [1].

Assembly101 [41] is the largest dataset among the three datasets with 4,321 videos and 202 coarse action classes composed of 11 verbs and 61 objects. Assembly101 is a procedural activity dataset containing videos of people assembling and disassembling 101 toy vehicles. On average, each video includes 24 action segments and is 7.1 minutes long. Compared to Breakfast, Assembly101 has 2.5 times more videos, 6.7 times more hours of video footage, 9.3 times more action segments, and 4.2 more action classes. For our evaluation, we follow the setting for temporal action segmentation [41] and report the results on the validation set.

For a fair comparison, we use the features that are provided for the datasets and that have been used in previous works. We use the I3D [7] features for the 50Salads and Breakfast datasets and TSM [29] features for the Assembly101 dataset [41]. Both features are 2048 dimensional vectors. Following [1], we also used the temporally downsampled features for 50Salads to compensate for the different frame-rates of the datasets.

Evaluation Metrics. We report the frame-wise accuracy (Acc), segmental Edit distance, and segmental F1 score at

the overlapping thresholds of 10%, 25%, and 50% denoted by $F1@\{10, 25, 50\}$. The intersection over the union (IoU) ratio is used as the overlapping threshold. The edit distance measures only the order of the actions but not the duration. The frame-wise accuracy measures the accuracy per frame. It is dominated by actions that have a long duration and it is not very sensitive to over-segmentation errors that occur when a few frames within a ground-truth action segment are wrongly classified. The F1 score is the most reliable measure.

4.1. How much temporal long-term context is needed?

We first present the results of our analysis on the impact of using the full sequence as input compared to using a temporal window. The goal of this experiment is to shed light on how much temporal long-term context is needed for the task of temporal action segmentation. For the analysis, we use only 50Salads and Assembly101 since the videos in Breakfast are too short. For the experiments, we train our approach (LTContext) and ASFormer [61] either on the full video sequences or we divide the videos into shorter sequences where the full context is lacking.

Fig. 4 shows the result of this experiment on the 50Salads and Assembly101 dataset. We report the window size as percentage of the average length of a video in the corresponding dataset and 100% means that the full sequence has been used. The results clearly show that the full context of an entire sequence is advantageous over allowing the model to see only a window of the input sequence even if the window is large (50%). We can also observe that our approach benefits more from the full sequence than ASFormer.

We furthermore evaluated whether the impact of the window size is stronger for longer videos. To this end, we sorted all test videos into four quarters by their length. Fig. 5 (left) shows that the difference between the window size 50% and the full video (100%) is larger for long videos. This shows that long-term context is in particular for long videos important. We also evaluated whether choosing a window for each video instead of choosing a window based on the average video length (fixed) performs better. For the video-specific window size, the window size is set to the percentage of each video. Fig. 5 (right) shows that a video-specific window size performs much worse than a fixed window. Varying the amount of context for each video is thus not beneficial. Fig. 1 shows qualitative results of our approach for different amounts of temporal context with a fixed window for a video from the Assembly101 dataset.

4.2. Comparison with State of the Art

We present the performance comparison of our method with state-of-the-art methods on the Breakfast and 50Salads datasets in Table 1 and on the Assembly101 dataset [41] in

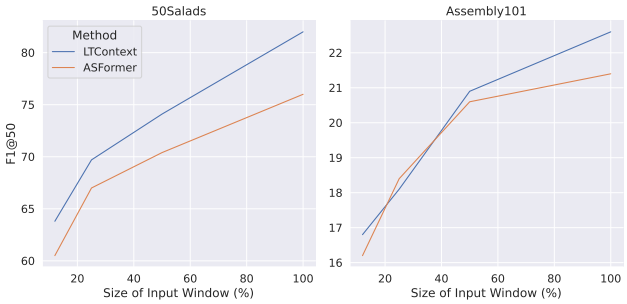

Figure 4: Impact of different sizes of the input window on the 50Salads dataset (left) and the Assembly101 dataset (right). The window size is given in percentage of the average length of a video in the corresponding dataset. 100% denotes entire video.

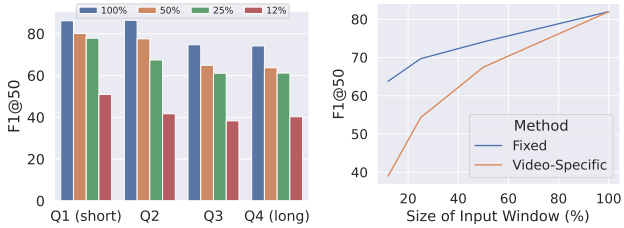

Figure 5: Results for different video lengths (rights). Comparison of a fixed with a video-specific input window size (left). Both plots are for 50Salads.

Table 2.

On Breakfast and 50Salads, our method outperforms all comparable methods in terms of F1 score at all thresholds, which is the most important measure. Our method also achieves a better Edit score than all methods except for UVAST [4] and TCTr [3]. UVAST uses ASFormer [61] as encoder and an additional alignment decoder. The higher Edit score of UVAST is expected since the approach has an additional head that predicts the sequence of actions and thus maximizes the Edit score by an additional loss. This, however, comes at the cost of a much lower frame accuracy. TCTr [3] achieves a higher frame-wise accuracy on Breakfast and a higher Edit score on 50 Salads, but it performs worse for the other metrics. The approach uses a boundary detection module and compresses the temporal features, which is a complementary approach. It needs to be noted that our approach outperforms TCTr for all metrics on 50Salads if we use 10 instead of 9 layers as shown in Table 9. However, 9 layers perform better than 10 layers for the other datasets. We therefore report in Tables 1 and 2 only the results with 9 layers.

We also compare to approaches that use different input features, perform additional test augmentation, or add additional constraints. On Breakfast, only DTL [59] reports better results. DTL [59] uses additional logic-based constraints for training ASFormer and assumes that action se-

Method	Note	Breakfast					50Salads				
		F1@{10, 25, 50}			Edit	Acc	F1@{10, 25, 50}			Edit	Acc
IDT+LM [39]	DF_1	-	-	-	-	-	44.4	38.9	27.8	45.8	48.7
ST-CNN [24]	DF_2	-	-	-	-	-	55.9	49.6	37.1	45.9	59.4
ED-TCN [23]	DF_2	-	-	-	-	-	68.0	63.9	52.6	59.8	64.7
TDRN [25]	DF_2	-	-	-	-	-	72.9	68.5	57.2	66.0	68.1
SSA-GAN [13]	DF_3	-	-	-	-	43.3	74.9	71.7	67.0	69.8	73.3
Bridge-Prompt [26]	DF_4	-	-	-	-	-	89.2	***87.8***	81.3	***83.8***	***88.1***
C2F-TCN [42]	TA	72.2	68.7	57.6	69.6	***76.0***	84.3	81.8	72.6	76.4	84.9
UVAST [4]+Viterbi	P	75.9	70.0	57.2	76.5	66.0	89.1	87.6	81.7	***83.9***	87.4
UVAST [4]+FIFA [43]	P	76.9	71.5	58.0	77.1	69.7	88.9	87.0	78.5	83.9	84.5
Liu [32] + ASRF [18]	P	77.5	72.3	59.5	76.7	73.7	87.9	86.6	80.5	82.7	86.6
DTL [59]	C	***78.8***	***74.5***	***62.9***	***77.7***	***75.8***	87.1	85.7	78.5	80.5	86.9
MS-TCN [1]	-	52.6	48.1	37.9	61.7	66.3	76.3	74.0	64.5	67.9	80.7
MS-TCN++ [27]	-	64.1	58.6	45.9	65.6	67.6	80.7	78.5	70.1	74.3	83.7
DTGRM [53]	-	68.7	61.9	46.6	68.9	68.3	79.1	75.9	66.1	72.0	80.0
MuCon [44]	-	73.2	66.1	48.4	76.3	62.8	-	-	-	-	-
Gao *et al*. [14]	-	74.9	69.0	55.2	73.3	70.7	80.3	78.0	69.8	73.4	82.2
BCN [56]	-	68.7	65.5	55.0	66.2	70.4	82.3	81.3	74.0	74.3	84.4
SSTDA [8]	-	75.0	69.1	55.2	73.7	70.2	83.0	81.5	73.8	75.8	82.2
C2F-TCN [42]	-	70.1	66.6	56.2	68.2	73.5	76.6	73.0	62.5	69.2	80.1
ASRF [18]	-	74.3	68.9	56.1	72.4	67.6	84.9	83.5	77.3	79.3	84.5
UVAST [4]	-	76.7	70.0	56.6	**77.2**	68.2	86.2	81.2	70.4	**83.9**	79.5
DPRN [35]	-	75.6	70.5	57.6	75.1	71.7	87.8	86.3	79.4	82.0	87.2
LGTNN [48]	-	76.2	71.5	57.5	75.2	72.5	87.5	86.2	79.8	82.0	86.1
ASFormer [61]	-	76.0	70.6	57.4	75.0	73.5	85.1	83.4	76.0	79.6	85.6
TCTr [3]	-	76.6	71.1	58.5	76.1	**77.5**	87.5	86.1	80.2	83.4	86.6
LTContext (Ours)	-	**77.6**	**72.6**	**60.1**	77.0	74.2	**89.4**	**87.7**	**82.0**	83.2	**87.7**

Table 1: Results on the Breakfast and 50Salads datasets. The best and second best results for the methods in the bottom half are shown in bold and underlined since only methods without additional notes are directly comparable. P: additional post-processing; C: additional constraints; DF: different features (DF_1: Improved Dense Trajectories (IDT), DF_2: Spatio-temporal VGG-style CNN, DF_3: Generative Adversarial Network (GAN), DF_4: ActionCLIP); TA: test augmentation. In the top half, we highlight results that are better in italic bold formatting.

Method	Assembly101						
	F1@{10, 25, 50}			Edit	Acc	Params (M)	Inference (sec)
MS-TCN++ [27]	31.6	27.8	20.6	30.7	37.1	1.08	0.1
UVAST [4]*	32.1	28.3	20.8	31.5	37.4	1.22	3.2
C2F-TCN [42]	33.3	29.0	21.3	**32.4**	39.2	6.89	0.1
ASFormer [61]*	33.4	29.2	21.4	30.5	38.8	1.13	2.1
LTContext (Ours)	**33.9**	**30.0**	**22.6**	30.4	**41.2**	0.72	0.6

Table 2: Comparison with state-of-the-art methods on the Assembly101 dataset. The best and second best results are shown in bold and underlined. *We trained UVAST [4] and ASFormer [61] using the code of the authors.

quences can be modeled by a directed acyclic graph. This is very efficient on the Breakfast dataset, which has relatively few actions per video compared to the other datasets. However, our approach outperforms DTL for all metrics on 50Salads where the structure of the action sequences is more complex. Furthermore, DTL cannot be applied to datasets like Assembly101 where actions occur more than once in a video. On 50Salads, only Bridge-Prompt [26] performs slightly better for some metrics. Bridge-Prompt proposes an approach for feature learning using a vision-language model such as ActionCLIP [55] in combination with ASFormer. The approach is thus complementary and not comparable to our approach. Nevertheless, our approach achieves higher $F1@10$ and $F1@50$ scores on 50Salads. If we use 10 instead of 9 layers as shown in Table 9, we even outperform Bridge-Prompt for all measures except for frame-wise accuracy. In summary, our approach outperforms all methods in terms of F1 score, which is the most precise measure of the segmentation quality, at all thresholds on 50Salads.

Assembly101 is the largest dataset both in terms of the number of videos and their length, and it is the most challenging dataset. Since ASFormer [61] and UVAST [4] have not been evaluated on this dataset, we trained ASFormer and UVAST on Assembly101 using the publicly available source code and report the results in Table 2 as well. We outperform all methods on Assembly101 in F1 scores. C2F-TCN [42] achieves the best edit score. Since we trained ASFormer [61] and UVAST [4] by ourselves, we can compare the training time on Assembly101. While our approach requires 1 day and 18 hours, ASFormer [61] and UVAST [4] needed 4 weeks and 2 weeks, respectively.

4.3. Qualitative Evaluation

In Figs. 6-8, we present some qualitative results for the Assembly101, Breakfast, and 50Salads dataset. The first row of each figure shows the middle frame of each ground-truth action segment. In the second, third, and fourth rows, the ground truth segmentation, the predictions of ASFormer [61], and the prediction of our model (LTContext) are shown, respectively. In Fig. 6, our approach shows much fewer errors than ASFormer. Although our approach recognizes all action classes that occur in the video, there are several errors where some instances are missed like the last action segment of the bottom row which corresponds to the action *'detach base'*. This indicates how challenging the Assembly101 dataset is. In Fig. 7, the predictions of our model are very close to the ground-truth. ASFormer overestimates the duration of the green segment, which corresponds to the action *'spoon powder'*, and hallucinates purple segments at the beginning and end of the video. In Fig. 8, both methods estimate the segments well, but ASFormer predicts wrongly two orange segments, which correspond to the action *'mix dressing'*, and the olive segment is too short, which corresponds to the action *'add salt'*.

4.4. Ablation Studies

We finally evaluate the impact of each component of our architecture. For the ablation studies, we report the results averaged over the 5 splits of the 50Salads dataset.

Impact of attention types In Table 3, we show the impact of using the combination of windowed and long-term context (LTContext) attention in the LTContext block illustrated in Fig. 3. We first compare it to two variants where we use only windowed or only long-term context attention. In order to keep the number of parameters the same, we still use

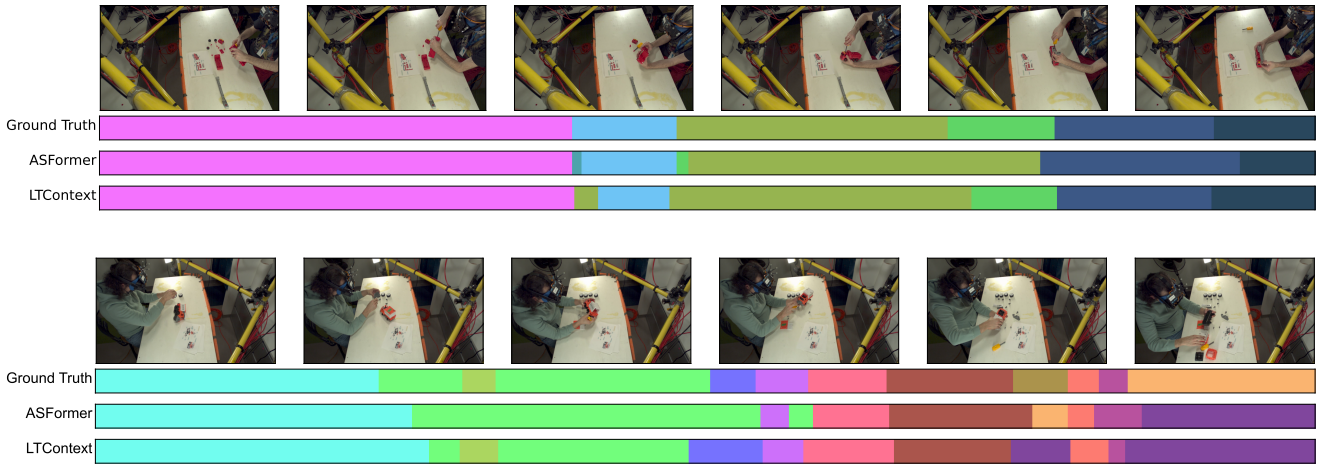

Figure 6: Qualitative results on Assembly101. The three rows show the ground-truth labels, the predictions by ASFormer, and the predictions by the proposed approach LTContext. It can be best viewed by using the zoom function of a PDF viewer.

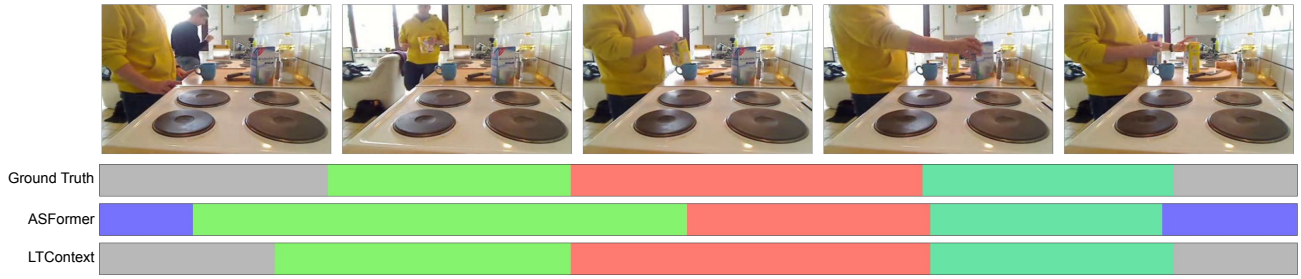

Figure 7: Qualitative results on Breakfast. The three rows show the ground-truth labels, the predictions by ASFormer, and the predictions by the proposed approach LTContext.

Model Architecture	F1@{10, 25, 50}			Edit	Acc
LTContext	**89.4**	**87.7**	**82.0**	**83.2**	**87.7**
- Windowed Attention	87.7	85.1	77.4	82.1	85.5
- LTContext Attention	84.1	82.6	74.1	78.2	85.3
WA (S1) + LTContext (S2-S4)	87.8	85.0	79.4	81.4	85.2

Table 3: Impact of using LTContext and Windowed Attention (WA) on 50Salads. In the case of Windowed Attention, we use two windowed attention blocks instead of a combination of windowed and LTContext attention. In the case of LTContext Attention, we use two LTContext attention blocks. The last row corresponds to using only windowed attention in stage 1 and only LTContext attention in stages 2 to 4.

in these cases two attention blocks within LTContext. The results show that combining both types of attention leads to better results. In particular, the F1@50 score is substantially higher.

As shown in Fig. 3, we use four LTContext blocks. We also evaluate what happens if we vary the attention not within an LTContext block but between the four LTContext blocks. For this, we use only windowed attention for the first LTContext block and only long-term context attention for the other three LTContext blocks. The last row in Table 3 shows that this does not perform better than using only windowed attention and it is worse than combining windowed and long-term context attention within an LTContext block.

Impact of different values of W and G The parameter W controls the size of the local window for the local attention and the parameter G controls the sparseness of the global attention. If not otherwise specified, we use $W = G = 64$ in our experiments. The results in Table 4 show that the F1 score drops when the size of the local window W becomes smaller. Note that larger values of W increase the memory and computational cost. When we decrease G, the F1 score also drops but not as drastic as for W.

Impact of overlaps for windowed attention As described in Section 3.1, we use an overlap for the keys and values for the windowed attention. In Table 5, we also report the result

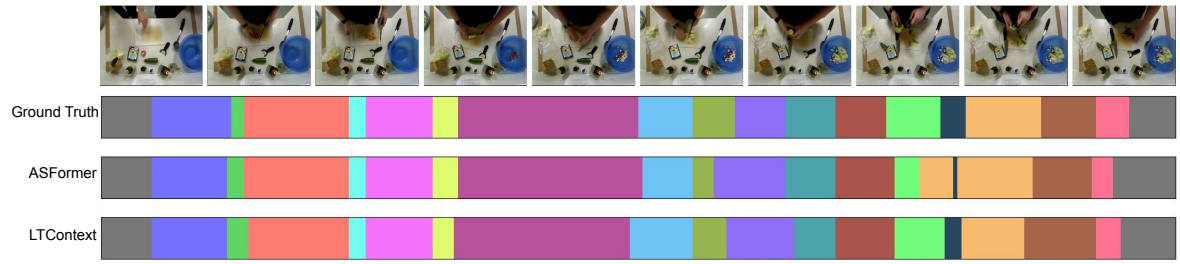

Figure 8: Qualitative results on 50Salads. The three rows show the ground-truth labels, the predictions by ASFormer, and the predictions by the proposed approach LTContext.

W	G	F1@{10, 25, 50}			Edit	Acc
8	64	83.8	82.1	74.5	77.0	87.0
16	64	85.4	82.9	77.1	77.1	87.0
32	64	87.6	85.8	79.8	81.5	86.6
64	64	**89.4**	**87.7**	**82.0**	**83.2**	**87.7**
64	32	88.8	87.2	81.3	83.2	87.7
64	16	88.6	87.0	80.3	82.6	86.2
64	8	88.6	87.1	79.9	82.9	86.6

Table 4: Impact of the parameters W and G on 50 Salads.

	F1@{10, 25, 50}			Edit	Acc
Non-Overlapping	87.9	85.5	79.1	81.4	85.1
Overlap (2 Windows)	**89.4**	87.7	**82.0**	**83.2**	**87.7**
Overlap (3 Windows)	89.2	**88.0**	80.8	**83.2**	86.6

Table 5: Impact of using an overlap for the keys and values for the local attention on 50Salads.

when the keys and values do not overlap, *i.e.*, they are the same as the queries. It is interesting to note that our attention can be interpreted as a combination of a reshaping operation with axial attention [16] in the special case without overlap. The results, however, show that an overlap improves the results. Using a larger overlap where the keys and values consist of three consecutive windows does not improve the results further.

Impact of the order of the attention The LTContext block shown in Fig. 3 uses first windowed attention and then long-term context attention. Table 6 shows the results when we change the order of windowed and long-term context attention within the LTContext block. If the order is changed, the performance decreases. Since the LTContext blocks are repeated, the drop in performance is moderate.

Impact of the cross-attention As described in Section 3.2, we use cross-attention in stages 2 to 4. Table 7 shows that the performance drastically decreases without cross-attention.

	F1@{10, 25, 50}			Edit	Acc
WA-LTContext	**89.4**	**87.7**	**82.0**	**83.2**	**87.7**
LTContext-WA	88.3	86.4	80.4	81.6	86.1

Table 6: Impact of the attention order within the LTContext block on 50Salads. Our model computes first windowed attention (WA) and then long-term context attention (row 1).

	F1@{10, 25, 50}			Edit	Acc
Without Cross-Attention	82.2	78.5	70.4	73.6	80.2
LTContext	**89.4**	**87.7**	**82.0**	**83.2**	**87.7**

Table 7: Impact of the cross-attention on 50Salads.

	F1@{10, 25, 50}			Edit	Acc
Conv with Dilation	**89.4**	**87.7**	**82.0**	**83.2**	**87.7**
Conv without Dilation	80.1	78.0	67.7	72.6	79.3
Without Conv	81.8	79.2	67.1	72.7	78.9

Table 8: Impact of using LTContext blocks with 1D convolution but without dilation (row 2) and using LTContext blocks without 1D convolution (row 3).

Impact of using convolutions The LTContext block shown in Fig. 3 starts with a dilated 1D convolution with kernel size 3. In Table 8, we evaluate the impact of the dilated convolution by comparing to a LTContext block that uses a convolutional kernel of the same size but without dilation factor, and a LTContext block without 1D convolution. The results show that dilated convolutions have a very high impact on the performance.

Impact of the number of layers As shown in Fig. 3, we repeat the LTContext blocks at each stage N times. We used N=9 layers in all experiments. Table 9 shows the impact of varying the number of layers. On 50Salads, all measures improve by increasing N. It needs to be noted that we use N=9 for the results reported in Table 1 although we can get even better results with 10 layers on 50 Salads. For the Breakfast and Assembly101 dataset, the best performance is

number of layers (N)	50Salads				
	F1@{10, 25, 50}			Edit	Acc
8	87.2	85.1	77.6	80.8	85.7
9	89.4	87.7	82.0	83.2	**87.7**
10	**89.5**	**88.1**	**82.4**	**84.1**	**87.7**
	Breakfast				
8	75.7	70.5	57.7	74.5	73.1
9	**77.6**	**72.6**	**60.1**	**77.0**	**74.2**
10	77.3	72.4	59.7	76.4	73.5
	Assembly101				
8	31.9	28.4	21.3	27.8	41.0
9	**33.9**	**30.0**	**22.6**	**30.4**	41.2
10	32.6	29.3	21.9	28.7	**41.5**

Table 9: Impact of the number of layers on 50Salads, Breakfast, and Assembly101.

number of heads	F1@{10, 25, 50}			Edit	Acc
1	**89.4**	**87.7**	**82.0**	**83.2**	**87.7**
2	88.8	87.0	80.5	82.9	87.0
4	89.2	87.1	80.8	83.1	86.9
8	88.2	86.4	80.5	82.1	86.1

Table 10: Impact of the number of attention heads.

achieved with 9 layers.

Impact of the number of attention heads In our implementation, we do not use multiple attention heads. Nevertheless, we evaluated the impact of using multiple heads in Table 10 since most transformers use multiple heads. The results, however, show that there is no benefit in using multiple heads. The results are consistent with the observations in [33, 30]. For example, [33] shows that many attention heads can often be reduced to a single head without impacting the performance. They also argue that some tasks are more reliant on multiple heads than others. Temporal action segmentation seems to be a task where one head is sufficient.

Impact of the number of stages As shown in Fig. 3, we use four stages of LTContext blocks, each of them with 9 layers. We evaluate the impact of the number of stages in Table 11. As can be seen, using multiple stages helps to reduce the over-segmentation error and improves the F1 score and Edit score significantly compared to using only one stage. Increasing the number of stages up to 4 improves all metrics, but using 5 stages decreases the performance and the network starts to overfit.

5. Conclusion

In this work, we addressed the question of how much temporal long-term context is needed for action segmentation.

number of stages	F1@{10, 25, 50}			Edit	Acc
1	56.3	54.1	49.9	45.6	84.7
2	86.2	84.5	78.2	80.5	86.7
3	87.6	85.7	78.3	81.2	85.5
4	**89.4**	**87.7**	**82.0**	**83.2**	**87.7**
5	89.0	86.8	80.0	83.0	85.7

Table 11: Impact of the number of stages on 50Salads.

Our analysis indicates that allowing networks to operate on the full input sequence is more beneficial compared to the case where the model has only access to a subset of the input. Based on our analysis, we presented LTContext, an approach for temporal action segmentation, where we leverage sparse attention to capture the long-term context of a video and windowed attention to model the local information in the neighboring frames. Our approach achieves state-of-the-art segmental F1 scores on the 50Salads and Assembly101 datasets, which contain long videos.

Acknowledgement

Juergen Gall has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) GA 1927/4-2 (FOR 2535 Anticipating Human Behavior), the project iBehave (receiving funding from the programme "Netzwerke 2021", an initiative of the Ministry of Culture and Science of the State of Northrhine Westphalia), and the ERC Consolidator Grant FORHUE (101044724). The sole responsibility for the content of this publication lies with the authors.

References

[1] Yazan Abu Farha and Juergen Gall. MS-TCN: Multi-stage temporal convolutional network for action segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A video vision transformer. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[3] Nicolas Aziere and Sinisa Todorovic. Multistage temporal convolution transformer for action segmentation. *Image and Vision Computing*, 2022.

[4] Nadine Behrmann, S. Alireza Golestaneh, Zico Kolter, Juergen Gall, and Mehdi Noroozi. Unified fully and timestamp supervised temporal action segmentation via sequence to sequence translation. In *European Conference on Computer Vision (ECCV)*, 2022.

[5] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document. *arXiv preprint arXiv:2004.05150*, 2020.

[6] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *International Conference on Machine Learning (ICML)*, 2021.

[7] 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)*, 2017.

[8] Min-Hung Chen, Baopu Li, Yingze Bao, Ghassan AlRegib, and Zsolt Kira. Action segmentation with joint self-supervised temporal domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[9] Zihang Dai, Hanxiao Liu, Quoc Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[10] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[11] 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. In *International Conference on Learning Representations (ICLR)*, 2021.

[12] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[13] Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Fine-grained action segmentation using the semi-supervised action gan. *Pattern Recognition*, 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] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. In *INTERSPEECH*, 2020.

[16] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv*, 2019.

[17] 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)*, 2020.

[18] 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)*, 2021.

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

[20] 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)*, 2014.

[21] 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.

[22] 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)*, 2016.

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

[24] 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)*, 2016.

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

[26] Muheng Li, Lei Chen, Yueqi Duan, Zhilan Hu, Jianjiang Feng, Jie Zhou, and Jiwen Lu. Bridge-prompt: Towards ordinal action understanding in instructional videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[27] Shi-Jie Li, Yazan Abu Farha, Yun Liu, Ming-Ming Cheng, and Juergen Gall. MS-TCN++: Multi-stage temporal convolutional network for action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[28] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[29] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[30] Liyuan Liu, Jialu Liu, and Jiawei Han. Multi-head or single-head? an empirical comparison for transformer training. *arXiv*, 2021.

[31] 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/CVF International Conference on Computer Vision (ICCV)*, 2021.

[32] Zhichao Liu, Leshan Wang, Desen Zhou, Jian Wang, Songyang Zhang, Yang Bai, Errui Ding, and Rui Fan. Temporal segment transformer for action segmentation. *arXiv preprint arXiv:2302.13074*, 2023.

[33] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[34] Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Łukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical transformers are more efficient language models. *arXiv preprint arXiv:2110.13711*, 2021.

[35] Junyong Park, Daekyum Kim, Sejoon Huh, and Sungho Jo. Maximization and restoration: Action segmentation through dilation passing and temporal reconstruction. *Pattern Recognition*, 2022.

[36] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning (ICML)*, 2018.

[37] Yifan Peng, Siddharth Dalmia, Ian Lane, and Shinji Watanabe. Branchformer: Parallel MLP-attention architectures to capture local and global context for speech recognition and understanding. In *International Conference on Machine Learning (ICML)*, 2022.

[38] Jiezhong Qiu, Hao Ma, Omer Levy, Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. Findings of Empirical Methods in Natural Language Processing (EMNLP), 2020.

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

[40] 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)*, 2012.

[41] F. Sener, D. Chatterjee, D. Shelepov, K. He, D. Singhania, R. Wang, and A. Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[42] Dipika Singhania, Rahul Rahaman, and Angela Yao. Coarse to fine multi-resolution temporal convolutional network. *arXiv preprint*, 2021.

[43] Yaser Souri, Yazan Abu Farha, Fabien Despinoy, Gianpiero Francesca, and Juergen Gall. Fifa: Fast inference approximation for action segmentation. In *DAGM German Conference on Pattern Recognition (GCPR)*, 2021.

[44] Yaser Souri, Mohsen Fayyaz, Luca Minciullo, Gianpiero Francesca, and Juergen Gall. Fast weakly supervised action segmentation using mutual consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

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

[46] 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)*, 2012.

[47] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Comput. Surv.*, 2022.

[48] Xiaoyan Tian, Ye Jin, and Xianglong Tang. Local–global transformer neural network for temporal action segmentation. *Multimedia Systems*, 2022.

[49] 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)*, 2021.

[50] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxim: Multi-axis mlp for image processing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[51] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *European Conference on Computer Vision (ECCV)*, 2022.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[53] Dong Wang, Di Hu, Xingjian Li, and Dejing Dou. Temporal relational modeling with self-supervision for action segmentation. *AAAI Conference on Artificial Intelligence*, 2020.

[54] 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.

[55] Mengmeng Wang, Jiazheng Xing, and Yong Liu. Actionclip: A new paradigm for video action recognition. *arXiv preprint arXiv:2109.08472*, 2021.

[56] 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.

[57] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[58] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollar, and Ross Girshick. Early convolutions help transformers see better. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[59] Ziwei Xu, Yogesh S Rawat, Yongkang Wong, Mohan Kankanhalli, and Mubarak Shah. Don't pour cereal into coffee: Differentiable temporal logic for temporal action segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[60] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal attention for long-range interactions in vision transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[61] Fangqiu Yi, Hongyu Wen, and Tingting Jiang. ASFormer: Transformer for action segmentation. In *British Machine Vision Conference (BMVC)*, 2021.

[62] Yanyi Zhang, Xinyu Li, Chunhui Liu, Bing Shuai, Yi Zhu, Biagio Brattoli, Hao Chen, Ivan Marsic, and Joseph Tighe. Vidtr: Video transformer without convolutions. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[63] Long Zhao, Zizhao Zhang, Ting Chen, and Dimitris Metaxas abd Han Zhang. Improved transformer for high-resolution GANs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.