Unified Fully and Timestamp Supervised Temporal Action Segmentation via Sequence to Sequence Translation
Supplementary Material

Nadine Behrmann[1,*], S. Alireza Golestaneh[1,*], Zico Kolter[1], Juergen Gall[2], and Mehdi Noroozi[1]

[1] Bosch Center for Artificial Intelligence
[2] University of Bonn, Germany

The structure of this supplementary material is as follows. In Section 1, we provide more details regarding the datasets that we are using as well as highlighting the main differences among them in terms of number of videos, lengths, and segments. In Section 2, we provide the implementation details of our architecture; we investigate the impact of the encoder architecture, our Split-Segment approach, and the constrained K-Medoids algorithm. Furthermore, we provide the values for our hyper-parameters. Last but not least, in Section 3, we provide more insight on our proposed grouping losses.

1 Datasets

In Table 1, we provide general information regarding the datasets (GTEA, 50Salads, and Breakfast) that we use for our experiments. Among these datasets Breakfast has the largest number of classes and videos. On the other hand, the GTEA and 50Salads dataset are much smaller in terms of number of samples. The 50Salads dataset has the longest video sequences among the three datasets, while GTEA contains videos with the highest number of action segments and repetitions of an action within a video.

Table 1. Statistics of Action Segmentation Datasets.

Dataset	# of Classes	# of Videos	Min Videos Length	Max Videos Length	Mean Videos Length	Min # of Segments in a Video	Max # of Segments in a Video
GTEA [1]	11	28	634	2009	1115.18	21	44
50Salads [4]	17	50	7555	18143	11551.90	15	26
Breakfast [2]	48	1712	130	9741	2097	2	25

2 Implementation Details

In Figure 1 we show the complete flowchart of our proposed model for the supervised setup. All of the training and testing experiments were conducted on a single NVIDIA V100 GPU.

* Equal contribution.

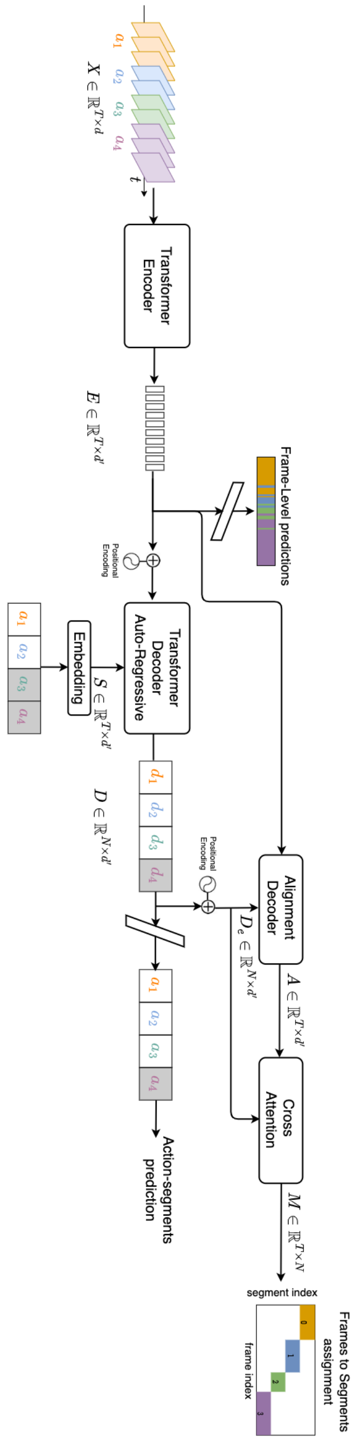
a_1 a_2 a_3 a_4
t
$X \in \mathbb{R}^{T \times d}$
Transformer Encoder
$E \in \mathbb{R}^{T \times d'}$
Frame-Level predictions
Positional Encoding
Transformer Decoder Auto-Regressive
Embedding
$S \in \mathbb{R}^{T \times d'}$
d_1 d_2 d_3 d_4
$D \in \mathbb{R}^{N \times d'}$
Alignment Decoder
$D_e \in \mathbb{R}^{N \times d'}$
$A \in \mathbb{R}^{T \times d'}$
Cross Attention
$M \in \mathbb{R}^{T \times N}$
Action-segments prediction
Frames to Segments assignment
segment index
frame index
0 1 2 3

Fig. 1. Flowchart of our proposed supervised action segmentation method

Impact of Encoder Model. In our proposed algorithm, we utilized a modified version of the encoder model proposed in [5]. The encoder model proposed by [5] takes advantage of window attention as well as hierarchical representation for the action segmentation task. While we find their proposed encoder model very effective and inspiring, we made small modifications to the architecture that further improved the performance. Particularly, we replace the `RELU` activation layers with `GELU` activation and add one more layer of dilated convolution at the end of each encoder block. Fig. 2 shows the side by side comparison between the encoder block proposed by [5] and our modified version. Moreover, Table 2 shows the Edit performance for using a simple encoder block (default pytorch implementation), the one proposed by [5], and our proposed modified version (last row in Table 2) on split 1 of GTEA, 50Salad, and Breakfast. While the ASFormer encoder achieves drastic improvements over the simple encoder, our modified version further improves over the ASFormer encoder.

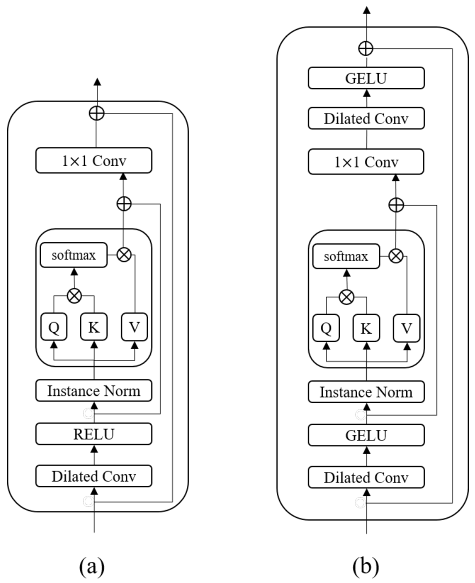

Fig. 2. Modifications on Encoder Model. Comparison between the original encoder block proposed by [5] (a) and our modified version (b) with `GELU` activations instead of `RELU` and an additional dilated convolution at the end of the encoder block.

Impact of Split-Segment. Due to a strong imbalance in the duration of action segments, we propose a *split-segment* approach for improving the training of the network, where longer action segments are split up into several shorter ones, so that segment durations are more uniformly distributed; it is important to note that during the inference we *do not* use any split-segment and use the video as is. In the timestamp supervised setting, we do not use split-segment as we do not have access to the ground truth du-

Table 2. Impact of Encoder Model. Quantitative comparison (Edit score) between the impact of different encoder blocks on split 1 of GTEA, 50Salads, and Breakfast.

		Breakfast	50Salads	GTEA
UVAST with	Simple Encoder	70.3	68.7	69.4
	ASFormer [5] Encoder	74.6	73.8	88.6
	Proposed Encoder	76.1	75.4	93.4
UVAST Timestamp	ASFormer [5] Encoder	72.9	76.6	89.0
	Proposed Encoder	74.3	78.3	89.1

ration of segments. For the split-segment approach, we scale the durations to $[0, 1]$ by dividing the absolute duration of a segment (*i.e.,* the number of frames in the segment) by the total number of frames in the video. For instance, if the split-segment value is set to 0.1 it means that the duration of each action segment should be at most 0.1 and segments larger than 0.1 will be split up into smaller segments with maximum length of 0.1. During inference we merge the repeated actions into one: For example, if our model predicts an action sequence of (A, B, B, C, A, A, A) we convert it to (A, B, C, A). The split-segment value is a hyper parameter that can be selected empirically for each dataset. In Table 3 we show the impact of using split-segment on split 1 of the Breakfast, 50Salads, and GTEA datasets, where we report the Edit scores. We can see that using the split-segment approach helps the model achieve better performances on all three datasets. Furthermore, in Table 4 we provide an ablation study regarding the impact of selecting different split-segment values on the split 1 of the Breakfast dataset.

Table 3. Impact of Split-Segment. Quantitative comparison (Edit score) between the impact of using split-segment versus not using it on split 1 of GTEA, 50Salads, and Breakfast dataset.

	Breakfast	50Salads	GTEA
No Split-Segment	75.0	74.2	88.2
With Split-Segment	76.1	75.4	93.4

Table 4. Impact of Split-Segment Values. Ablation study on split 1 of the Breakfast dataset regarding the impact of using different values for the split-segment on the performance (Edit score).

	Split-Segment Values					
	0.05	0.1	0.15	0.17	0.2	0.3
Breakfast	74.9	75.6	76.1	76.1	76.1	75.4

Hyper-Parameters. Table 5 provides a summary of the values for our hyper-parameters used for training our model. In Table 5, Stage 1 and 2 refer to Section 3.1 and 3.3 of the main paper, where we train the Transformer for auto-regressive segment prediction and alignment Transformer for duration prediction, respectively. Furthermore, as shown in Table 5, we use a cross-attention smoothing (average pooling on the cross-attention map along the T dimension) for the 50Salads dataset to reduce the noise in the cross-attention. Comparing to the Breakfast and GTEA datasets, 50Salads dataset has the longest videos (see Table 1) with a very low number of training data which causes the cross-attention map to be noisy. Our proposed cross-attention loss along with cross-attention smoothing helps to reduce the noise and therefore leads to a better performance. Similar to previous methods and to have a fair comparison, we use sampling rate of 2 for the 50Salads dataset since it has higher FPS compared with the other two datasets [3]. We state the hyper-parameters used in FIFA and Viterbi in Table 6.

Table 5. Hyper-Parameters.

	Common Between Stage 1 and 2									Stage 1 (Encoder-Decoder)						Stage 2 (Alignment Decoder)				
	Batch Size	Optimizer	LR	Epoch	sampling rate	d	d’	τ'	Dropout	Activation	Split Segment	# Layers in Encodeer	# Layers in Decoder	Decoder Feedforward Dimension	Cross-Attention Smoothing	Smoothing Kernel	# Parameters	# Layers in Alignment Decoder	Decoder Feedforward Dimension	# Parameters
Breakfast	1	adam	0.0005	800	1	2048	64	0.001	✓	GELU	0.17	10	2	2048	×	×	1.109M	1	1024	0.166M
50Salads	1	adam	0.0005	800	2	2048	64	0.001	✓	GELU	0.15	10	2	2048	✓	31	1.103M	1	1024	0.166M
GTEA	1	adam	0.0005	800	1	2048	64	0.001	✓	GELU	0.17	10	2	2048	×	×	1.102M	1	1024	0.166M

Table 6. Hyper-Parameters used in FIFA and Viterbi.

Dataset	FIFA			Viterbi
	Epochs	Sharpness	Step-size	frame sampling
Breakfast	3000	80	0.01	5
50Salads	3000	80	0.01	2
GTEA	3000	80	0.1	1

K-Medoids. In the main paper, we propose a constrained k-medoids algorithm for generating pseudo-segmentations given frame-wise input features and timestamps. In contrast to the vanilla k-medoids clustering algorithm, our constrained version ensures temporal consistency of the clusters and the resulting temporally continuous clusters can be unambiguously identified with the class labels of the ground truth transcript. This is a major advantage over an unconstrained clustering method, which may result in temporally fragmented clusters, making class label assignment ambiguous. We compare our constrained k-medoids with the unconstrained version, see Table 7, where we assign each cluster the class label belonging to the timestamp it was initialized with. Note, that in this scenario the original ground truth timestamps may end up in completely different clusters and the class label assignment becomes noisy.

As expected the temporal fragmentation of the clusters leads to over-segmentation and correspondingly low Edit and F1 scores. However, even on Acc this unconstrained version suffers due to the noisy label assignment. Furthermore, we show two example videos from the Breakfast dataset in Fig. 3; again, we observe over-segmentation due to temporally fragmented clusters. Notably, we observe that the unconstrained k-medoids performs much better on GTEA compared with Breakfast and 50Salads. One reason for this can be the frequency of background classes, which are visually distinct to the action classes in the video and typically show very static scenes. The frequent background classes of highly similar features separate the action classes from one another. In contrast, Breakfast and 50Salads do not have a frequent background class and relatively static scenes are assigned to an action class, making it more difficult to separate them.

3 Grouping Loss Terms

The action segmentation datasets mentioned above are highly imbalanced in terms of the frequency of the actions that appear in the videos and the number of frames each

Table 7. K-Medoids. We compare our constrained k-medoids algorithm proposed in the main paper with the vanilla unconstrained version.

	Breakfast					50Salads					GTEA				
	F1@					F1@					F1@				
	{10	25	50}	Edit	Acc	{10	25	50}	Edit	Acc	{10	25	50}	Edit	Acc
Constrained k-medoids	95.5	87.5	70.0	100.0	76.9	97.5	90.4	75.6	100.0	81.3	99.8	97.7	83.0	100.0	75.3
Unconstrained k-medoids	8.4	6.6	3.8	12.3	53.8	3.8	2.5	1.0	2.3	52.9	71.0	68.2	52.9	59.8	69.6

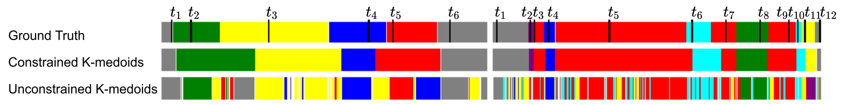

Fig. 3. K-Medoids. We compare our constrained and the unconstrained k-medoids algorithm qualitatively. Both algorithms cluster the frame-wise input features, using the ground truth timestamps $t_1, \ldots, t_n$ as initialization.

action occupies, which influences the cross-entropy loss on top of the encoder and decoder. To cope with the imbalanced classes, we utilize two modified versions of the cross-entropy loss. The first loss involves averaging the probabilities of each class separately, and the second one involves averaging the logits of each class before passing them to `Softmax`.

To shed more light on the intuition behind the modifications, let us consider a classifier that classifies N frames x_n into C classes. We denote the logits by a_n and the corresponding probabilities by $\mu_n = \texttt{Softmax}(a_n)$ where $\texttt{Softmax}(a_n)_c = \frac{e^{a_{n,c}}}{\sum_{i=1}^{C} e^{a_{n,i}}}$. For grouping the frames by class label, we define:

$$N_c = \{n | y_n = c\} \quad \text{for } c \in \{1, \ldots, C\}, \tag{1}$$

$$\bar{\mu}_c = \frac{1}{|N_c|} \sum_{n \in N_c} \mu_n, \tag{2}$$

$$\hat{\mu}_c = \texttt{Softmax}(\hat{a})_c \quad \hat{a}_c = \frac{1}{|N_c|} \sum_{n \in N_c} a_{n,c}. \tag{3}$$

We consider the following three loss terms, where the first is an element-wise cross-entropy loss and the last two group-wise cross-entropy loss terms, taking the average outside and inside the `Softmax`, respectively:

$$\mathcal{L} = -\sum_{n=1}^{N} \sum_{c=1}^{C} y_{n_c} \log \mu_{n_c}, \tag{4}$$

$$\bar{\mathcal{L}} = -\sum_{c=1}^{C} \log \bar{\mu}_c, \tag{5}$$

$$\hat{\mathcal{L}} = -\sum_{c=1}^{C} \log \hat{\mu}_c. \tag{6}$$

Table 8 summarizes the results for different choices of the grouping loss. We observe that $\bar{\mathcal{L}}$ works the best for the segment-wise loss on top of the decoder, and $\hat{\mathcal{L}}$ works best for the frame-wise classification. Note that we report the results of the first stage training, *i.e.,* transcript prediction only and therefore only report Edit score.

Table 8. Impact of different modifications on the group loss. Ablation study on split 1 of the 50Salads dataset regarding the impact of using different modifications of the group loss. We report Edit results for stage 1 training.

$\mathcal{L}_{\text{g-frame}}$	$\mathcal{L}_{\text{g-segment}}$	50Salads
$\mathcal{L}$, Eq. (5)	$\mathcal{L}$, Eq. (5)	78.4
$\hat{\mathcal{L}}$, Eq. (6)	$\hat{\mathcal{L}}$, Eq. (6)	78.1
$\bar{\mathcal{L}}$, Eq. (5)	$\hat{\mathcal{L}}$, Eq. (6)	79.8

References

1. Fathi, A., Ren, X., Rehg, J.M.: Learning to recognize objects in egocentric activities. In: CVPR (2011) 1
2. Kuehne, H., Arslan, A., Serre, T.: The language of actions: Recovering the syntax and semantics of goal-directed human activities. In: CVPR (2014) 1
3. Li, S.J., Abu Farha, Y., Liu, Y., Cheng, M.M., Gall, J.: MS-TCN++: Multi-stage temporal convolutional network for action segmentation. TPAMI (2020) 5
4. Stein, S., McKenna, S.J.: Combining embedded accelerometers with computer vision for recognizing food preparation activities. In: Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing (2013) 1
5. Yi, F., Wen, H., Jiang, T.: ASFormer: Transformer for action segmentation. In: BMVC (2021) 3, 4