TAEC: Unsupervised Action Segmentation with Temporal-Aware Embedding and Clustering Supplementary

1. Introduction

For additional insights into TAEC, we introduce the background of spectral clustering in Sec. 2.1 and give details of the Viterbi decoding in Sec. 2.2. We perform more ablation studies on comparing baseline embeddings and clustering methods (Sec. 3.1), scaling of spatio-temporal similarity (Sec. 3.2), cluster ordering (Sec. 3.3), decoding strategies (Sec. 3.4). Finally, we provide more quantitative (Sec. 3.5) and qualitative segmentation results (Sec. 3.6) on the three datasets.

2. Method

2.1. Spectral Clustering

Background information related to Sec. 3.2.1 in the main manuscript: Given the embedded feature sequence $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_T$, we build a frame-to-frame similarity graph $G \in \mathbb{R}^{T\times T}$, whose edge weight $g(i,j),\ i,j \in \{1,...,T\}$ represents the similarity between frame i and frame j. Grouping the frames into K clusters can be interpreted as a graph partition problem by cutting edges on G, resulting in K subgraphs $G_1, G_2, ..., G_K$. The normalized cut (Ncut) problem [1] is employed to compute a balanced partition by minimizing the energy

$$\mathcal{L}_{cut}(G_1, G_2, ..., G_K) = \frac{1}{2}\sum_{k=1}^{K}\frac{W(G_k, \overline{G}_k)}{\mathrm{vol}(G_k)}, \quad (1)$$

where $W(G_k, \overline{G}_k)$ represents the sum of edge weights between elements in the subgraph G_k and elements of all the other subgraphs, *i.e.*, the sum of weights of edges to be cut. $\mathrm{vol}(G_k)$ is the sum of weights of edges within the resulting subgraph G_k. Spectral clustering [2] is a relaxed solution to this NP-hard minimization problem in Eq. (1) and has shown good performance on many graph-based clustering problems, *e.g.* [3, 4, 5]. Note that while K-means operates on Euclidean distance in the feature space and assumes convex and isotropic clusters, spectral clustering can find clusters with non-convex boundaries.

2.2. Frame Labeling by Viterbi Decoding

Additional explanations to Sec. 3.3 in the main manuscript: The global cluster assignment delivers the ordered clusters on each video, which are aligned across all videos. To compute the final segmentation, we use the resulting ordering and decode each video into a sequence of K temporally consistent segments. That is, we determine the optimal label sequence $\hat{y}_{1\sim T_n,n} = \{y_{1,n}, ..., y_{T_n,n}\}$ by re-assigning each frame to one of the temporally ordered clusters.

Given the embedded feature sequence $\mathbf{e}_{1\sim T_n,n} = \{\mathbf{e}_{1,n}, ..., \mathbf{e}_{T_n,n}\}$ and the temporal order of the clusters, we search for the optimal label sequence that maximizes the probability $p(y_{1\sim T_n,n}|\mathbf{e}_{1\sim T_n,n})$. Following [6], this posterior probability can be factorized into the product of likelihoods and the probability of a given temporal order, *i.e.*,

$$\begin{aligned}\hat{y}_{1\sim T_n,n} &= \underset{y_{1\sim T_n,n}}{\arg\max}\ p(y_{1\sim T_n,n}|\mathbf{e}_{1\sim T_n,n}) \\ &= \underset{y_{1\sim T_n,n}}{\arg\max}\ \{\Pi_{t=1}^{T_n} p(\mathbf{e}_{t,n}|y_{t,n}) \cdot \Pi_{t=1}^{T_n} p_n(y_{t,n}|y_{1\sim(t-1),n})\} \\ &= \underset{y_{1\sim T_n,n}}{\arg\max}\ \{\Pi_{t=1}^{T_n} p(\mathbf{e}_{t,n}|y_{t,n}) \cdot p_n(y_{t,n}|y_{t-1,n})\} \quad (2)\end{aligned}$$

Here the likelihood $p(\mathbf{e}_{t,n}|y_{t,n})$ is the probability of a frame embedding $\mathbf{e}_{t,n}$ from the video n belonging to a cluster. Therefore, we fit a Gaussian distribution on each global cluster and compute the frame-wise likelihoods with the Gaussian model, *i.e.*,

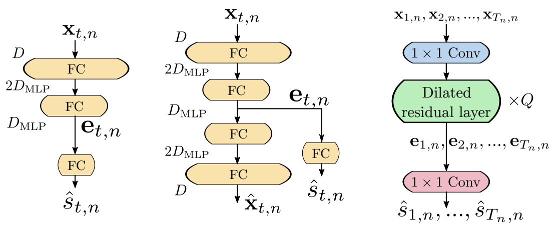
$$p(\mathbf{x}|k) = \mathcal{N}_k(\mathbf{x}; \mu_k, \Sigma_k), k \in \{1, ..., K\}. \quad (3)$$

$p_n(y_{t,n}|y_{t-1,n})$ is the transition probability from label $y_{t-1,n}$ at frame $t-1$ to label $y_{t,n}$ at frame t, which is defined by the temporal order of clusters. We denote the set of frame transitions defined by the temporal order of clusters on the n-th video by O_n, *e.g.*, for the temporal order of $a \to b \to c \to d$, $O_n = \{a \to b, b \to c, c \to d\}$. The transition probability is binary, *i.e.*,

$$\begin{aligned}&p_n(y_{i,n}|y_{i-1,n}) \quad (4)\\ &\quad = \mathbb{1}(y_{i,n} = y_{i-1,n} \vee y_{i-1,n} \to y_{i,n} \in O_n).\end{aligned}$$

This means that we allow either a transition to the next cluster according to the temporal order, or we keep the cluster assignment of the previous frame.

26th Computer Vision Winter Workshop, Robert Sablatnig and Florian Kleber (eds.), Krems, Lower Austria, Austria, Feb. 15-17, 2023

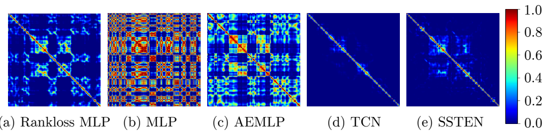


(a) MLP (b) AEMLP (c) TCN

Figure 1: Three baseline variants of embedding networks.

Table 1
Comparison of combinations of embeddings and clustering methods on Breakfast (in %).

Model		K-means			Two-step clustering		
Feature	Embedding	MoF	IoU	F1	MoF	IoU	F1
	Rankloss [7]	35.2	15.6	28.8	34.7	13.4	23.7
	MLP	**42.9**	13.1	25.5	32.7	10.9	21.2
DTFV	AEMLP	34.7	13.6	25.8	32.6	11.6	21.4
	TCN	33.4	**17.8**	31.3	40.4	**19.1**	32.9
	SSTEN	39.3	**17.8**	**31.9**	**50.3**	19.0	**33.6**



(a) Rankloss MLP (b) MLP (c) AEMLP (d) TCN (e) SSTEN

Figure 2: Frame-to-frame similarity matrices of the embedded sequences of the same video (computed by different embedding networks from DTFV features) on Breakfast.

Note that in **two-step clustering**, we derive the temporal order of clusters on each video separately, by sorting the clusters on the video according to the average timestamp. Therefore, we have an individual O_n for each video n. On the contrary, in **K-means**, there is a uniform order of global clusters for all the videos and O_n is thus the same for each video n.

The Viterbi algorithm for solving Eq. (2) is performed in an iterative process using dynamic programming, *i.e.*,

$$p(y_{1\sim t,n}|\mathbf{e}_{1\sim t,n}) = \max_{y_{t,n}} \{p(y_{1\sim t-1,n}|\mathbf{e}_{1\sim t-1,n}) \cdot p(\mathbf{e}_{t,n}|y_{t,n}) \cdot p(y_{t,n}|y_{t-1,n})\}. \tag{5}$$

The sequences that do not follow the temporal order will be filtered out in an early stage to narrow down the search range for the optimal label sequence. The output of the Viterbi decoding is the optimal cluster label sequence, *i.e.*, $\hat{y}_{1\sim T_n,n}$.

3. Additional Results

3.1. Embedding and clustering

Further, we compare our SSTEN embedding with three baseline variants (shown in Fig. 1): MLP temporal embedding, autoencoder with MLP (AEMLP) and temporal convolutional network (TCN), in combination with the two clustering methods.

MLP uses three FC layers for relative timestamp prediction. *AEMLP* uses MLP-based autencoder for both relative timestamp prediction and feature reconstruction. *TCN* deploys Q stacked dilated residual layers only for relative timestamp prediction.

Here, we also implement the Rankloss MLP embedding [7] for reference. We report the performance of these five embeddings in Table 1.

Comparison of the five embeddings. We learn the five embeddings (Rankloss MLP, MLP, AEMLP, TCN and SSTEN) on the DTFV features. Here, the Rankloss MLP (consisting of two FC layers) is trained with a ranking loss. We use the initialization of uniform segmentation as the temporal prior to train the model with only one iteration.

TCN and SSTEN are both networks for sequence-to-sequence learning, while Rankloss MLP, MLP and AEMLP are trained on individual frames. By comparing the performance between these two groups in Table 1, we see that sequence-to-sequence learning leads to better performance, especially when combined with the two-step clustering, which results in clusters with better temporal consistency.

For the two-step clustering, we also plot the frame-to-frame similarity matrices (spatial Gaussian kernel) of the five embeddings for the same Breakfast video in Fig. 2. The plots show that Rankloss MLP, MLP and AEMLP, which are trained on individual frames, do not expose an appropriate temporal structure. There are noisy block patterns even in positions far away from the diagonal, which results in noisy clusters and thus, leads to erroneous temporal orders and inferior assignment results in the two-step clustering. The least noisy Rankloss MLP has the highest performance among these three. On the contrary, TCN and SSTEN embedded features, which show a clear diagonal block structure in the similarity graph, achieve a better performance in the two-step clustering. This verifies that the sequence-to-sequence embedding learning (TCN and SSTEN) and two-step clustering are a well-suited combination to address the sequential nature of frames in both processing steps of feature embedding and clustering.

Considering K-means clustering, the merit of having a better sequential nature of the embedded features via sequence-to-sequence learning can also be seen from the higher IoU and F1 scores (TCN: IoU 17.8%/F1 31.3%, *vs.* SSTEN: 17.8%/31.9%), as these penalize dominating segments and oversegmentation.

In contrast to TCN, SSTEN can preserve the spatial layout of the input features due to the feature reconstruction via the autoencoder. By comparing TCN and SSTEN, we see that the SSTEN embedding with feature reconstruction leads to a boost in the MoF score. The marginal improvement of AEMLP over MLP is due to the fact that the MLP structure with only FC layers is not well-suited for feature reconstruction.

Comparison between K-means and two-step clustering. Considering the performance of the five embeddings with the two clustering methods, we see that K-means leads to higher scores on the inferior embeddings (Rankloss MLP, MLP and AEMLP) trained on individual frames, while two-step clustering performs better on sequence-to-sequence learning-based embeddings (TCN and SSTEN). When combined with the proposed SSTEN embedding, two-step clustering outperforms K-means by a large margin in terms of the MoF score. We also tried applying K-means on each video separately. However, the performance dropped significantly. K-means depends only on the spatial distance and results in oversegmentation, which leads to erroneous temporal order on each video and thus, an inferior global cluster assignment.

3.2. Impact of Scaling in Spatiotemporal Similarity

We perform spectral clustering with the proposed spatiotemporal similarity. Here, we analyze the impact of the scaling factors in the spatial and temporal Gaussian kernels, *i.e.*, σ^2_{spat} and σ^2_{tmp}. These adjust the extent to which two frames are considered similar to each other and influence the clustering quality. The experiments are conducted for SSTEN embeddings on Breakfast.

Impact of the scaling of the spatial Gaussian kernel. For local scaling, we set $\sigma^2_{\text{spat}} = \sigma_i \sigma_j$, where σ_i is the distance from $\mathbf{e}_i$ to its m-th nearest neighbor in the feature space. The resulting segmentation performance w.r.t. m is shown in Fig. 3. With m varying in the range of 3 to 20, the IoU and F1 scores remain stable. There is a range of $m \in \{8, 9\}$ where the best MoF scores are achieved, whereas for other scaling parameters, the MoF score drops. Thus, we set $m = 9$ for all following evaluations.

For comparison, we also set σ_{spat} to fixed values (without local scaling) and report the segmentation performance in Table 2. We achieve great results at smaller σ_{spat} values (0.5 and 0.7).

However, with increasing σ_{spat} the MoF score drops significantly, while there are only minor fluctuations in IoU and F1. Apparently, σ_{spat} has a large impact on the clustering quality. The local scaling eases the effort of tuning σ_{spat} by dynamically determining the scaling factor.

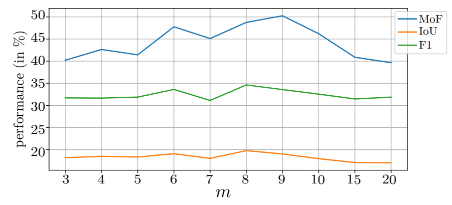


Figure 3: Segmentation performance of the two-step clustering on SSTEN embeddings ($\lambda = 0.002$) with different m (for local scaling) on Breakfast.

Table 2
Segmentation performance of two-step clustering on SSTEN embeddings ($\lambda = 0.002$) with respect to a fixed spatial scaling factor σ_{spat} (without local scaling) on Breakfast (in %).

σ_{spat}	MoF	IoU	F1
0.5	47.1	18.0	33.4
0.7	**48.0**	18.5	**33.9**
1	41.1	**19.2**	32.9
2	38.4	18.6	32.2
10	35.0	17.9	31.2

Impact of the scaling of the temporal Gaussian kernel. The temporal Gaussian kernel is operated on the temporal distance between frames in a video. With $\sigma^2_{\text{tmp}} = 2\sigma'^2$, the term $\exp\left(-(s_i - s_j)^2/(2\sigma'^2)\right)$ is in the standard form of a Gaussian kernel. We set $\sigma' = 1/6$ so that the $6\sigma'$ range of the temporal Gaussian is equal to the video length (since the length of each video is normalized to 1 for the relative timestamp prediction). The segmentation performance with respect to σ' is shown in Table 3. Apparently, $\sigma' = 1/6$ leads to the best result. Here,

Table 3
Segmentation performance of two-step clustering on SSTEN embeddings ($\lambda = 0.002$) with respect to the temporal scaling factor σ' ($\sigma^2_{\text{tmp}} = 2\sigma'^2$ on Breakfast (in %).

σ' ($\sigma^2_{\text{tmp}} = 2\sigma'^2$)	MoF	IoU	F1
∞ (w/o tmp. Gauss)	41.5	16.5	30.6
1/3	43.5	16.9	31.3
1/6	**50.3**	**19.0**	33.6
1/12	44.3	18.5	**34.1**

we also evaluate the case without the temporal Gaussian kernel, which leads to a drop in performance. The impact of the temporal Gaussian kernel on similarity matrices of SSTEN embeddings can also be seen by comparing the top and bottom rows in Fig. 6 in the main manuscript. For example, by adding the temporal Gaussian kernel, we decrease the similarities in Fig. 6(a1) according to the temporal distance between two frames, which leads to clearer diagonal block structure in Fig. 6(a2). Thus, we set $\sigma' = 1/6$ for all following evaluations.

Table 4
Impact of cluster order for two-step clustering on SSTEN embeddings (in %).

Order	Breakfast				YTI			
	MoF	IoU	F1	Edit	MoF	IoU	F1	Edit
video-wise	50.3	**19.0**	**33.6**	**42.3**	**46.6**	**10.7**	**29.5**	**25.5**
uniform	**53.5**	15.7	32.2	33.0	40.7	7.7	25.1	20.3

3.3. Impact of Cluster Order

One merit of performing within-video clustering is that we can derive the temporal order of sub-clusters for each video separately. The video-wise individual order of clusters is used to guide the Viterbi decoding, which breaks the common assumption that clusters follow the same temporal order in all the videos. In the following, we verify the efficacy of the derived video-wise order of clusters. We use the same within-video clustering result with global cluster assignment and perform Viterbi decoding using two different temporal cluster orders: (1) *video-wise* order: the temporal order of sub-clusters is determined on each video separately; and (2) *uniform* order: the uniform order is determined by sorting the average timestamps of global clusters and is then applied to all the videos. Table 4 reports the segmentation performance (after Viterbi) with these two orders for our SSTEN embeddings on Breakfast and YTI. To measure the correctness of the predicted segment order, we adopt the segmental edit distance (Edit), which is a common metric for supervised action segmentation, *e.g.*, [8, 9, 10, 11]. It penalizes segmentation results that have a different segment order than the ground truth (*i.e.*, it penalizes out-of-order predictions, as well as oversegmentation).

From Table 4 we see that our video-wise order clearly outperforms the uniform order except for MoF on Breakfast. Furthermore, the edit score verifies that our derived video-wise temporal orders are valid.

In our experiments we especially notice that the MoF and IoU scores could act contradictory to each other, *e.g.*, the uniform order results in higher MoF scores (on Breakfast) at the cost of lower IoU scores. MoF tends to overfit on dominant classes (*e.g.*, classes with longer action instances) while IoU is sensitive to underrepresented classes and penalizes segmentation results with dominating segments. Therefore, it is necessary that we consider all metrics for evaluation, as a higher MoF score does not always correspond to better performance in practice.

3.4. Impact of Decoding Strategies

We compare our approach, which uses Viterbi decoding, with the Mallow model decoding that has been proposed in [7]. The authors propose a Rankloss embedding over all video frames from the same activity with respect to a pseudo ground truth action annotation. The embedded frames of the whole activity set are then clustered and the likelihood for each frame and for each cluster is computed. For the decoding, the authors build a histogram of features with respect to their clusters with a hard assignment and set the length of each action with respect to the overall amount of features per bin. After that, they apply a Mallow model to sample different orderings for each video with respect to the sampled distribution. The resulting model is a combination of Mallow model sampling and action length estimation based on the frame distribution.

For this experiment, we evaluate the impact of the different decoding strategies on two embeddings: the Rankloss embedding [7] and our SSTEN embedding. Table 5 reports the results of these two embeddings in combination with three decodings: the Mallow model, Viterbi decoding with K-means and Viterbi decoding with two-step clustering.

Table 5
Comparison of combinations of embeddings and decoding strategies on Breakfast (in %).

Decoding	Rankloss [7] embed.			SSTEN embed.		
	MoF	IoU	F1	MoF	IoU	F1
Mallow [7]	34.7	**17.8**	**31.4**	36.4	18.1	31.5
kmeans+Viterbi	**35.2**	15.6	28.8	39.3	17.8	31.9
two-step.+Viterbi	34.7	13.4	23.7	**50.3**	**19.0**	**33.6**

Following [7], we run 7 iterations for the Rankloss embedding with the Mallow model. In each iteration, the Rankloss embedding is retrained using the segmentation result from the last iteration as pseudo label, and the frame-wise likelihoods and the Mallow model are updated.

Unlike the Mallow model, our Viterbi decoding is a one-iteration procedure. It is operated on the embedding which is trained only once. When combining with Viterbi, we train the Rankloss model only once using the initialized uniform segmentation as a prior. For SSTEN with the Mallow model, we only run for one iteration, as we do not need to train SSTEN with pseudo labels iteratively.

Considering the Rankloss results in Table 5 we see that combining it with the Mallow model achieves its highest IoU and F1 scores. This is because for Viterbi decoding, the Rankloss model trained only one-time using the uniform initialization as pseudo label is lacking of a strong temporal prior. Considering SSTEN, our Viterbi decoding with two-step clustering clearly outperforms the Mallow model. With Mallow, the SSTEN embedding has competitive IoU and F1 scores but significantly lower MoF. We also tried running the Mallow model on SSTEN embedded features for multiple iterations. However, this resulted in a reduced number of clusters. Thus, we see that an appropriate combination of embedding and decoding strategy is necessary.

To have a closer look into the Viterbi decoding, we visualize the likelihood grids computed from global clusters, as well as the resulting decoding path over time for two videos on Breakfast in Fig. 4. It shows that the decoding, which generates a full sequence of actions, is able to marginalize actions that do not occur in the video by just assigning only very few frames to those ones and the majority of frames are assigned to the clusters that occur in the video. Even if the given temporal order constrains that the resulting K coherent segments have to follow the fixed temporal order, the segments that actually do not belong in the sequence will be marginalized because the Viterbi algorithm decodes a path that maximizes the posterior probability. Overall, it turns out that the Viterbi decoding constrained by a temporal order performs better than the Mallow model's iterative re-ordering.

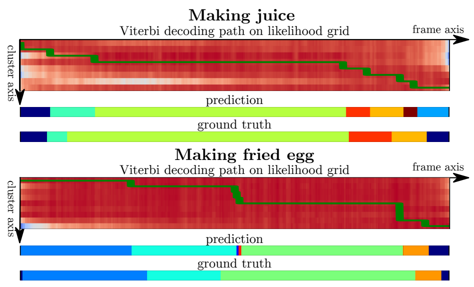


Figure 4: Comparison of Viterbi decoding path on the likelihood grid computed from the global clusters resulted from two-step clustering on the SSTEN embeddings, for two videos on Breakfast. The warm (red)/cool (blue) colors in the grid indicate high/low likelihoods of a frame belonging to an action class. It shows that the decoding assigns most frames to occurring actions while marginalizing actions that do not occur in the sequence by assigning only a few frames.

3.5. Quantitative Segmentation Results

3.5.1. Results of Clustering and Final Segmentation

In order to show the advantage of two-step clustering over K-means, when combined with the proposed SSTEN embedding, we report both, the results of clustering (before Viterbi decoding) and the final segmentation performance (after Viterbi decoding) on Breakfast in Table 6. We see that the proposed two-step clustering leads to superior performance than K-means, in terms of both clustering (before Viterbi decoding) in most metrics, and in terms of final segmentation (after Viterbi decoding).

Table 6
Comparison of combinations of SSTEN and different clustering methods in terms of clustering and final segmentation after Viterbi decoding on Breakfast (in %).

Embedding	Clustering	Clustering results			Final results		
		MoF	IoU	F1	MoF	IoU	F1
SSTEN	K-means	27.2	13.5	**26.3**	39.3	17.8	31.9
	two-step.cluster	**38.6**	**13.7**	25.9	**50.3**	**19.0**	**33.6**

3.5.2. Segmentation Results On Each Activity

We report the ground truth number of classes and segmentation performance of MLP with K-means (*MLP+kmeans*, our reimplementation of [12]) and TAEC for each activity on Breakfast (Table 7), YTI (Table 8) and 50 Salads (Table 9). The evaluation is done with global Hungarian matching on all videos. The number of clusters is set to the maximum number of ground truth classes for each activity (K = max.#gt).

Table 7
Maximum number of ground truth action classes and segmentation performance of MLP+kmeans (our reimplementation of [12]) and TAEC for the 10 activities on Breakfast (in %). The number of clusters is set to the maximum number of ground truth classes for each activity (K = max.#gt).

Breakfast					
Global Hungarian matching on all videos (K = max.#gt)					
Activity	K	Methods	MoF	IoU	F1
coffee	7	MLP+kmeans	**46.8**	**15.7**	**26.2**
		TAEC	35.6	15.2	24.9
cereals	5	MLP+kmeans	48.8	25.8	37.2
		TAEC	**59.0**	**31.4**	**47.7**
tea	7	MLP+kmeans	32.2	13.0	22.7
		TAEC	**39.2**	**16.3**	**26.1**
milk	5	MLP+kmeans	40.4	21.2	36.6
		TAEC	**46.7**	**27.3**	**43.5**
juice	8	MLP+kmeans	36.9	14.1	27.9
		TAEC	**52.2**	**22.3**	**36.2**
sandwich	9	MLP+kmeans	47.4	15.0	25.3
		TAEC	**53.7**	**19.8**	**33.5**
scrambled egg	12	MLP+kmeans	34.5	10.8	19.9
		TAEC	**48.1**	**15.2**	**28.3**
fried egg	9	MLP+kmeans	36.4	11.5	24.5
		TAEC	**49.1**	**17.4**	**30.5**
salad	8	MLP+kmeans	34.7	7.8	27.5
		TAEC	**42.0**	**15.2**	**34.0**
pancake	14	MLP+kmeans	57.4	8.6	19.2
		TAEC	**58.2**	**19.2**	**35.8**
All	-	MLP+kmeans	42.9	13.1	25.5
		TAEC	**50.3**	**19.0**	**33.6**

Table 8
Maximum number of ground truth action classes and the segmentation performance of MLP+kmeans (our reimplementation of [12]) and TAEC for the 5 activities on YTI (in %). The number of clusters is set to the maximum number of ground truth classes for each activity (K = max.#gt).

YouTube Instructions							
Global Hungarian matching on all videos (K = max.#gt)							
Activity	K	Methods	MoF w/o bkg	IoU w/o bkg	F1 w/o bkg	MoF w bkg	IoU w bkg
coffee	10	MLP+kmeans	40.9	10.4	**34.2**	11.9	9.5
		TAEC	**42.8**	**10.5**	26.6	**12.4**	**9.6**
change tire	11	MLP+kmeans	45.9	17.2	34.0	24.7	15.8
		TAEC	**58.6**	**20.0**	**37.2**	**31.5**	**18.4**
jump car	12	MLP+kmeans	30.6	4.5	24.4	5.1	4.1
		TAEC	**34.3**	**6.2**	**26.4**	**5.7**	**5.8**
cpr	7	MLP+kmeans	34.4	**9.9**	31.2	15.0	**8.6**
		TAEC	**38.0**	7.9	**33.3**	**16.6**	6.9
repot	8	MLP+kmeans	29.8	**7.2**	**23.1**	10.1	**6.4**
		TAEC	**35.8**	7.1	22.4	**12.1**	6.3
All	-	MLP+kmeans	39.4	9.9	**29.6**	14.4	9.7
		TAEC	**46.6**	**10.7**	29.5	**17.0**	**10.5**

Table 9
Maximum number of ground truth action classes and the segmentation performance of MLP+kmeans (our reimplementation of [12]) and TAEC for the single activity on 50 Salads (in %). The number of clusters is set to the maximum number of ground truth classes for each activity (K = max.#gt).

50 Salads					
Global Hungarian matching on all videos (K = max.#gt)					
Activity	K	Methods	MoF	IoU	F1
salad	eval 12	MLP+kmeans	37.9	24.6	40.2
		TAEC	**48.4**	**26.0**	**44.8**
	mid 19	MLP+kmeans	**29.1**	**15.7**	**23.4**
		TAEC	26.6	14.9	**23.4**

3.6. Qualitative Segmentation Results

We show the qualitative results of clustering and final segmentation on 7 composite activities: *making cereals* (Fig. 5), *making milk* (Fig. 6), *making juice* (Fig. 7), *making fried egg* (Fig. 8), *making pancake* (Fig. 9) on Breakfast, *changing tire* (Fig. 10) on YTI and *making salad* (Fig. 11) on 50 Salads (eval 12 classes). The mapping between cluster labels and ground truth classes is done with global Hungarian matching on all videos. The number of clusters is set to the maximum number of ground truth classes for each activity (K = max.#gt).

For each activity, we visualize the results of 10 videos. For each video, the 3-row-group displays the ground truth (1st row), TAEC (2nd row), MLP+kmeans [12] (3rd row).

References

[1] J. Shi, J. Malik, Normalized cuts and image segmentation, PAMI 22 (2000) 888–905.
[2] U. V. Luxburg, A tutorial on spectral clustering, Statistics and Computing 17 (2007) 395–416.
[3] H.-C. Huang, Y.-Y. Chuang, C.-S. Chen, Affinity aggregation for spectral clustering, in: CVPR, 2012.
[4] Z. Li, J. Chen, Superpixel segmentation using linear spectral clustering, in: CVPR, 2015.
[5] R. Liu, H. Zhang, Segmentation of 3d meshes through spectral clustering, in: Pacific Conference on Computer Graphics and Applications, 2004.
[6] A. Richard, H. Kuehne, A. Iqbal, J. Gall, NeuralNetwork-Viterbi: A framework for weakly supervised video learning, in: CVPR, 2018.
[7] F. Sener, A. Yao, Unsupervised learning and segmentation of complex activities from video, in: CVPR, 2018.
[8] Y. A. Farha, J. Gall, MS-TCN: Multi-stage temporal convolutional network for action segmentation, in: CVPR, 2019.
[9] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, G. D. Hager, Temporal convolutional networks for action segmentation and detection, in: CVPR, 2017.
[10] V. I. Morariu, L. S. Davis, Multi-agent event recognition in structured scenarios, in: CVPR, 2011.
[11] H. S. Koppula, R. Gupta, A. Saxena, Learning human activities and object affordances from rgb-d videos, The International Journal of Robotics Research 32 (2013) 951–970.
[12] A. Kukleva, H. Kuehne, F. Sener, J. Gall, Unsupervised learning of action classes with continuous temporal embedding, in: CVPR, 2019.

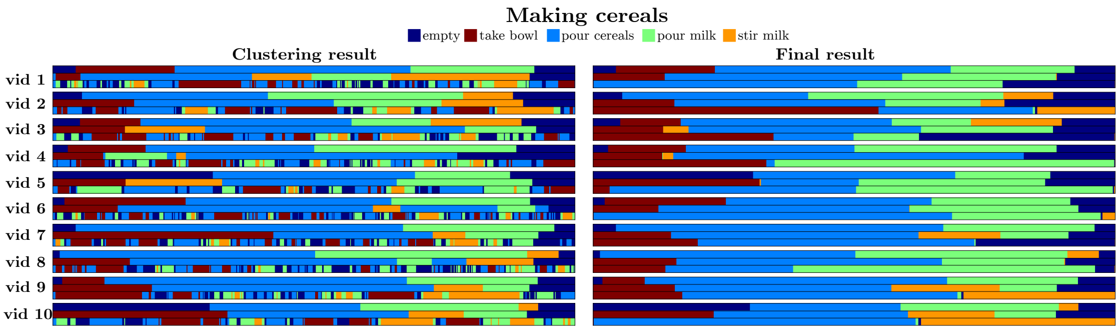


Figure 5: Qualitative results of clustering and final segmentation of 10 *making cereals* videos on Breakfast. For each video, the 3-row-group displays the ground truth (1st row), TAEC (2nd row), MLP+kmeans [12] (3rd row). The mapping between cluster labels and ground truth classes is done with global Hungarian matching on all videos. The number of clusters is set to the maximum number of ground truth classes for each activity (K = max.#gt).

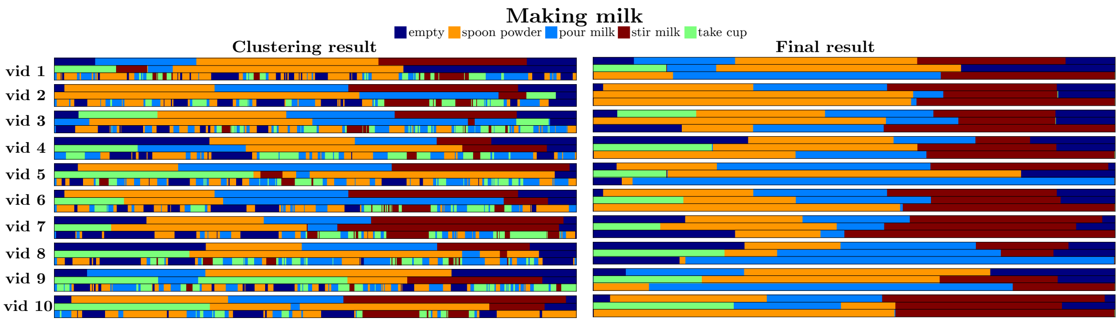


Figure 6: Qualitative results for 10 *making milk* videos on Breakfast.

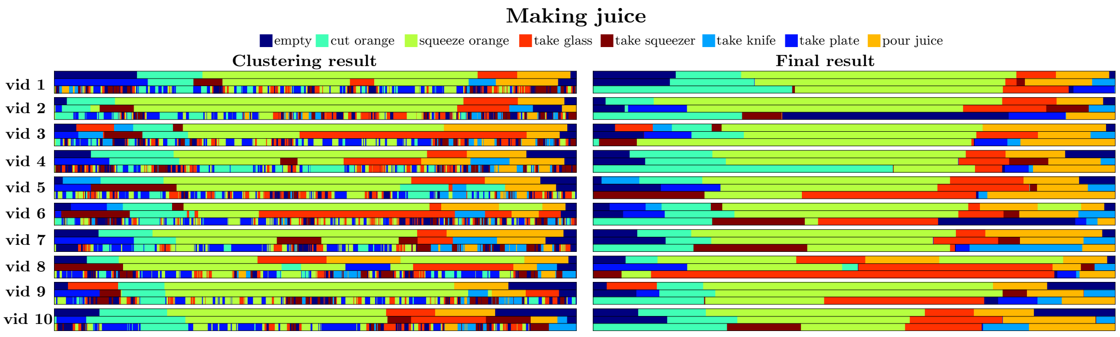


Figure 7: Qualitative results for 10 *making juice* videos on Breakfast.

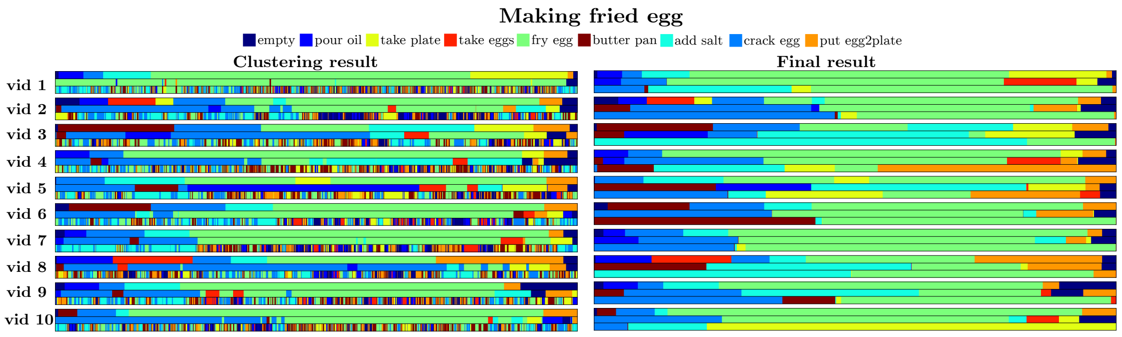


Figure 8: Qualitative results for 10 *making fried egg* videos on Breakfast.

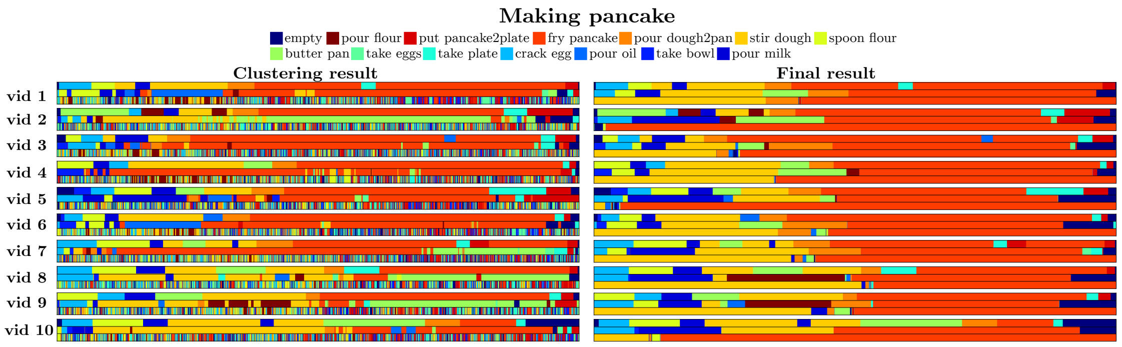


Figure 9: Qualitative results for 10 *making pancake* videos on Breakfast.

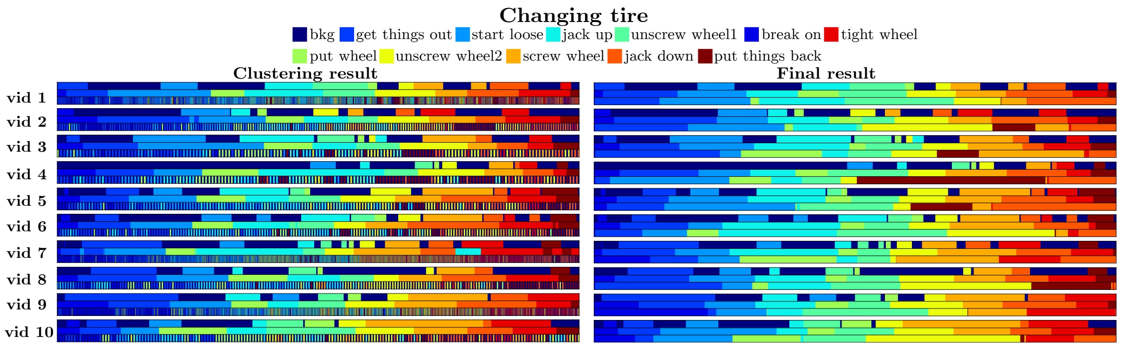


Figure 10: Qualitative results for 10 *changing tire* videos of the YouTube Instructions dataset. Similar to the Breakfast illustrations, for each video the 3-row-group shows the ground truth (1st row), TAEC (2nd row), and MLP+kmeans [12] (3rd row).

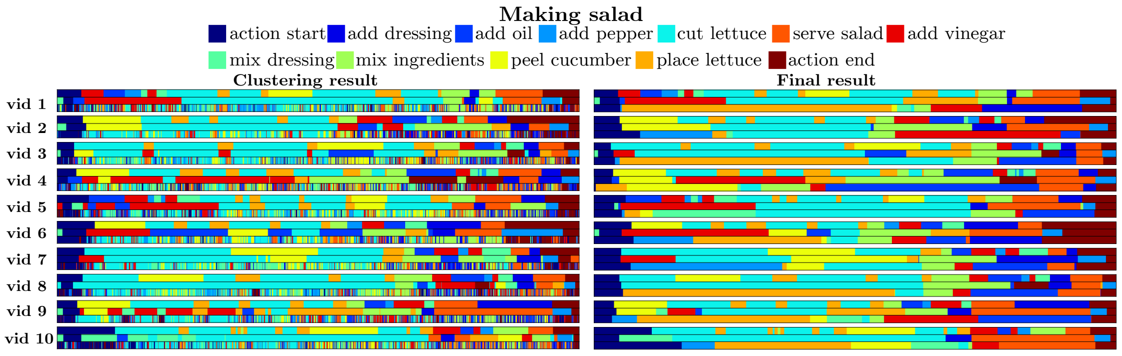


Figure 11: Qualitative results for 10 *making salad* videos on the 50 Salads dataset (from the *eval*-level, 12 action classes). Again, for each video the 3-row-group shows the ground truth (1st row), TAEC (2nd row), and MLP+kmeans [12] (3rd row).