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

Wei Lin[1,2], Anna Kukleva[3], Horst Possegger[1], Hilde Kuehne[4] and Horst Bischof[1]

[1]Institute of Computer Graphics and Vision, Graz University of Technology, Austria
[2]Christian Doppler Laboratory for Semantic 3D Computer Vision, Austria
[3]Max-Planck-Institute for Informatics, Germany
[4]Goethe University Frankfurt, Germany

Abstract

Temporal action segmentation in untrimmed videos has gained increased attention recently. However, annotating action classes and frame-wise boundaries is extremely time consuming and cost intensive, especially on large-scale datasets. To address this issue, we propose an unsupervised approach for learning action classes from untrimmed video sequences. In particular, we propose a temporal embedding network that combines relative time prediction, feature reconstruction, and sequence-to-sequence learning, to preserve the spatial layout and sequential nature of the video features. A two-step clustering pipeline on these embedded feature representations then allows us to enforce temporal consistency within, as well as across videos. Based on the identified clusters, we decode the video into coherent temporal segments that correspond to semantically meaningful action classes. Our evaluation on three challenging datasets shows the impact of each component and, furthermore, demonstrates our state-of-the-art unsupervised action segmentation results.

Keywords

Unsupervised learning, unsupervised clustering, action segmentation

1. Introduction

Action recognition has seen tremendous success in recent years, especially in the context of short video clip classification [1, 2, 3], action detection [4, 5, 6], and temporal action segmentation [7, 8, 9, 10]. The top-performing methods for temporal action segmentation, however, require frame-wise annotations, which is expensive and impractical for large-scale datasets [7, 8, 9, 11]. Consequently, a large body of research focuses on weakly-supervised approaches where only an ordered list [12, 13, 14, 15, 16] or an unordered set [17, 18, 19] of action labels is needed. These approaches assume that the actions that occur in each training video are known, and sometimes even require their exact ordering. Acquiring such ordered action lists, however, can still be time consuming or even infeasible. For applications like indexing large video datasets or human behavior analysis in neuroscience or medicine, it is often unclear what actions should be annotated. In these cases, it is necessary to automatically discover and identify recurring actions in large video datasets.

To address this problem, Sener and Yao [20] proposed the task of unsupervised action segmentation to identify patterns of recurring actions in long, untrimmed video sequences that correspond to semantically meaningful action classes. Recent approaches for unsupervised action segmentation, *e.g.* [20, 21, 22], focus on three aspects: (1) finding a suitable embedding space for the video data, (2) identifying clusters of temporal segments across a large amount of videos, and (3) parsing the input videos given the respective feature embedding and cluster information. Sener and Yao [20] tackle the action segmentation task with a linear embedding and Mallow decoding, while other approaches follow the pipeline of an MLP [21] or U-Net embedding [22], K-means clustering, and Viterbi decoding. However, these methods do not fully leverage the temporal relationships of frames within a video, either neglecting this information when learning the embedding [20, 22] or when clustering [21, 22].

Since temporal consistency is essential for all steps of the action segmentation pipeline, we propose TAEC, an approach that considers the sequential nature of frames in a video for both, embedding learning and clustering. The main steps of our approach are illustrated in Fig. 1.

Specifically, we first propose a sequence-to-sequence temporal embedding network (SSTEN) that combines pretext tasks of relative timestamp prediction and autoencoder feature reconstruction. While the autoencoder reconstruction retains the feature layout, the relative timestamp prediction encodes the relative temporal information within a video. Sequence-to-sequence learning enables the embedding of spatial layout and temporal information on a complete video sequence.

To cluster the embedded features, we propose a temporal-aware two-step clustering approach that con-

26th Computer Vision Winter Workshop, Robert Sablatnig and Florian Kleber (eds.), Krems, Lower Austria, Austria, Feb. 15-17, 2023
wei.lin@icg.tugraz.at (W. Lin); akukleva@mpi-inf.mpg.de (A. Kukleva); possegger@icg.tugraz.at (H. Possegger); kuehne@uni-frankfurt.de (H. Kuehne); bischof@icg.tugraz.at (H. Bischof)

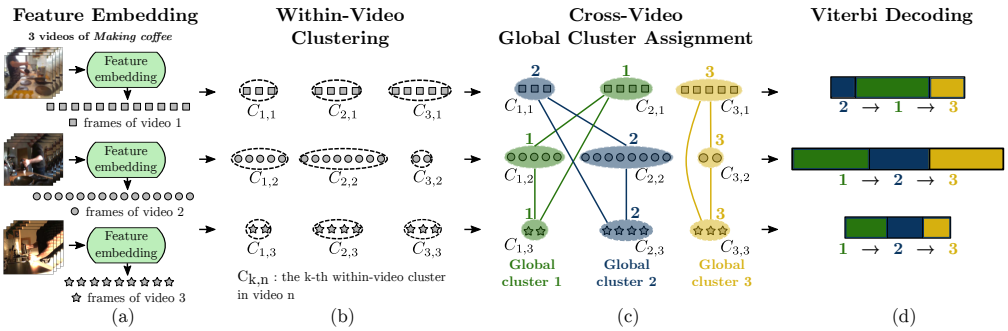


Figure 1: Pipeline of TAEC. We compute the embedded features with the sequence-to-sequence temporal embedding network (SSTEN). Thereupon, we perform a within-video clustering on each video individually and apply a cross-video global cluster assignment to group the within-video clusters into global clusters. The global cluster assignment also defines the ordering of the clusters in each video. Finally, we use Viterbi decoding to estimate temporally coherent segments for each video.

sists of a within-video clustering and a cross-video global cluster assignment. Specifically, we perform clustering within each video, with a spatio-temporal similarity among frames. Then we conduct global cluster assignment to group the clusters across videos. The global cluster assignment defines the ordering of the clusters for each video. In this way, we overcome the unrealistic assumption that actions of an activity always follow the same temporal order. Such an assumption is commonly used in related works, *e.g.* [21, 22]. For instance, in the activity of *making coffee*, a unified temporal order between actions such as *adding milk* and *adding sugar* is assumed for all videos of *making coffee*, whereas our approach can handle changes of the action order in different videos. After assigning all within-video clusters to a set of global clusters, we perform Viterbi decoding to obtain a segmentation of temporally coherent segments.

Our contributions can summarized as following:

- We design a sequence-to-sequence temporal embedding network (SSTEN), which combines relative timestamp prediction, autoencoder reconstruction and sequence-to-sequence learning.
- We propose a within-video clustering with a novel spatio-temporal similarity formulation among frames.
- We propose a cross-video global cluster assignment to group within-video clusters across videos into global clusters, which also overcomes the assumption that in all videos of an activity, actions follow the same temporal order.

2. Related Work

Unsupervised learning of video representations is commonly performed via pretext tasks, such as reconstruction [23, 24], future frame prediction [22, 25, 26], and recognition of frame orders [27, 28, 29, 30, 31]. For instance, Srivastava *et al.* [24] exploit an LSTM-based autoencoder for learning video representations. Villegas *et al.* [26] and Denton and Birodkar [25] employed two encoders to generate feature representations of content and motion. The temporal order of frames or small chunks is utilized as a self-supervision signal for representation learning on short video clips in [27] and [28]. Inspired by these approaches, we employ two self-supervision tasks: feature reconstruction and relative time prediction.

Clustering of temporal sequences has been explored for parsing human motions [32, 33, 34, 35]. While Zhang *et al.* [35] proposed a hierarchical dynamic clustering framework, Li *et al.* [33] and Tierney *et al.* [34] explored temporal subspace clustering to segment human motion data. In contrast to unsupervised action segmentation, these methods are applied on each temporal sequence individually and do not consider association among sequences. Instead, we propose a cross-video global cluster assignment to group within-video clusters across different videos into global clusters.

Unsupervised action segmentation on fine-grained activities has recent work that either focus on the representation learning [20, 22, 36] or the clustering step [23]. However, the temporal information is neglected in at least one of these two steps. For representation learning, Sener and Yao [20] construct a feature embedding by learning a linear mapping from visual features to a latent space with a ranking loss. However, the linear model trained with individual frames does not consider the temporal association between frames. VidalMata *et al.* [22] employ a U-Net trained on individual frames for future frame prediction. Predicting for one or a few steps ahead only requires temporal relations within a small temporal window. Instead, we propose to learn a representation by predicting the complete sequence of relative timestamps to encode the long-range temporal information.

For the clustering step, related works [22, 23] neglect temporal consistency of frames within a video. Instead, we apply within-video clustering on each video with a proposed similarity formulation that considers both spatial and temporal distances.

Two recent approaches perform clustering [37] or cluster-agnostic boundary detection [38] on each video separately, without identifying clusters or segments across videos. [37] solves a task similar to human motion parsing and evaluates the segmentation for each video individually. [38] only detects boundaries of category-agnostic segments, and does not identify if some segments within a video or across videos are of the same category. On the contrary, our segments on all videos are category-aware as they are aligned globally across videos by our global cluster assignment.

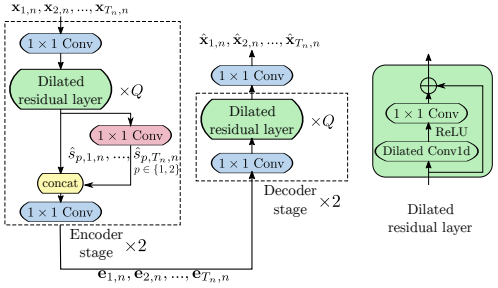


Figure 2: Architecture of SSTEN: We stack 2 stages of encoder and decoder for sequence-to-sequence feature reconstruction. Each stage consists of Q dilated residual layers with dilated temporal convolution. The intermediate representation in encoder is used for relative time prediction (red block).

3. Temporal-Aware Embedding and Clustering (TAEC)

We address unsupervised action segmentation as illustrated in Fig. 1. First, we learn a suitable feature embedding (Sec. 3.1). We then perform within-video clustering on each video (Sec. 3.2.1), and group the within-video clusters into global clusters (Sec. 3.2.2). Finally, we compute temporally coherent segments on each video using Viterbi decoding (Sec. 3.3).

3.1. SSTEN: Sequence-to-Sequence Temporal Embedding Network

To learn a latent representation for temporal sequences, we adopt a sequence-to-sequence autoencoder. Inspired by the multi-stage temporal convolutional network [7], we use a concatenation of two stages for both encoder and decoder, as shown in Fig. 2. Given a set $\{\mathbf{X}_n\}_{n=1}^{N}$ of N videos, where each video $\mathbf{X}_n = \{\mathbf{x}_{t,n}\}_{t=1}^{T_n}$ has T_n frames, the outputs are reconstructed frame features $\{\hat{\mathbf{x}}_{t,n}\}_{t=1}^{T_n}$. The embedded features are the hidden representation $\{\mathbf{e}_{t,n}\}_{t=1}^{T_n}$.

Every encoder and decoder stage consist of 1×1 convolution layers for dimension adjustment (Fig. 2 blue) and Q dilated residual layers (green), each containing a dilated temporal 1D convolution. Since no fully connected layers are employed, sequences of variable lengths can be processed seamlessly. The dilation rate at the q-th layer is 2^{q-1}. By stacking dilated residual layers, the temporal receptive field increases exponentially. The receptive field of the q-th layer is $1 + (r-1) \times (2^q - 1)$, where r is the kernel size. Therefore, each frame in the hidden representation has a long temporal dependency on the input video. In each encoder stage, we use a 1×1 convolution layer (in red) to predict the frame-wise relative timestamps $s_{t,n} = \frac{t}{T_n}$. At the end of each encoder stage, the hidden representation is a concatenation (in yellow) of the features from dilated residual layers and the predicted relative timestamps. The training loss is:

$$\mathcal{L} = \lambda \sum_{n=1}^{N} \sum_{t=1}^{T_n} \|\mathbf{x}_{t,n} - \hat{\mathbf{x}}_{t,n}\|_2^2 + \sum_{p \in \{1,2\}} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (s_{t,n} - \hat{s}_{p,t,n})^2, \tag{1}$$

where the coefficient λ balances the two terms. The pretext tasks of reconstruction and relative timestamp prediction encode both, the spatial distribution and the global temporal information, into the embedded features. We compare SSTEN with several baseline embedding networks in the supplementary.

3.2. Two-Step Clustering

After learning the feature embedding, we group the embedded features into K clusters by a within-video clustering and a cross-video global cluster assignment.

3.2.1. Within-Video Clustering

We perform spectral clustering on frames within each video (detailed description in the supplementary). Given the embedded feature sequence[1] $[\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_T]$, we build a frame-to-frame similarity matrix $G \in \mathbb{R}^{T \times T}$. The entries $g(i,j),\ i,j \in \{1, ..., T\}$, represent the similarity between frame i and frame j. To consider both the spatial and temporal distance of features, we propose to measure the similarity by the product of two Gaussian kernels

$$g(i,j) = \exp\left(-\frac{\|\mathbf{e}_i - \mathbf{e}_j\|_2^2}{\sigma_{\text{spat}}^2}\right) \cdot \exp\left(-\frac{(s_i - s_j)^2}{\sigma_{\text{tmp}}^2}\right), \tag{2}$$

where s_i, s_j are the corresponding relative timestamps of frame i, j and $\sigma_{\text{spat}}, \sigma_{\text{tmp}}$ are the scaling factors for the

[1]For ease of notation, we omit the video index n.

spatial and temporal Gaussian kernels. To avoid manually tuning σ_{spat}, we use local scaling [39] to estimate σ_{spat} dynamically. To this end, we replace σ^2_{spat} by $\sigma_i\sigma_j$, where σ_i is the distance from $\mathbf{e}_i$ to its m-th nearest neighbor in the embedding space. We provide an ablation study on scaling of the spatio-temporal similarity in the supplementary. Consequently, frames of similar visual content and relative timestamps are encouraged to be grouped into the same cluster.

3.2.2. Cross-Video Global Cluster Assignment

After within-video clustering, we assign the $N \times K$ within-video clusters across videos into K global clusters. Every global cluster should contain N within-video clusters, each coming from a different video (*c.f.*, Fig. 1). This can be interpreted as an N-dimensional assignment problem [40].

We regard the n-th video $V_n = \{\mathbf{c}_{k,n} | k = 1, .., K\}$ as a vertex set, where each k-th within-video cluster $\mathbf{c}_{k,n}$ is a vertex. We construct an N-partite graph $G = (V_1 \cup V_2 \cup ... \cup V_N, E)$. $E = \bigcup_{m<n, m,n \in \{1,...,N\}} \{(\mathbf{c}, \mathbf{c}') | \mathbf{c} \in V_m, \mathbf{c}' \in V_n\}$ is the set of edges between within-video clusters across videos. The edge weight $w(\mathbf{c}, \mathbf{c}')$ is the distance between centroids of two within-video clusters $\mathbf{c}, \mathbf{c}'$. The solution to the N-dimensional assignment is a partition by dividing the graph G into K cliques $Z_1, Z_2, ..., Z_K$. A clique Z_k, which is a subset of N vertices from N different vertex sets, defines the k-th global cluster. The induced sub-graphs of the cliques $Z_1, Z_2, ..., Z_K$ are complete and disjoint. We denote the edge set of the induced sub-graph of Z_k as E_{Z_k}. The cost of a clique is the sum of pairwise edge weights between the contained vertices. The cost of an assignment solution is the sum of the costs of all the K cliques, *i.e.*,

$$\mathcal{L}(Z_1, Z_2, ..., Z_K) = \sum_{k=1}^{K} \sum_{(\mathbf{c},\mathbf{c}') \in E_{Z_k}} w(\mathbf{c}, \mathbf{c}'). \quad (3)$$

In order to solve this NP-hard problem, we employ an iterative multiple-hub heuristic [41]. In each iteration, we choose a hub vertex set $V_h = \{\mathbf{c}_{k,h} | k = 1, .., K\}$ and there are $(N-1)$ non-hub vertex sets. We compute an assignment solution in each iteration in two steps, as is shown in Fig. 3: (1) We first perform $(N-1)$ bipartite matchings between V_h and each of the remaining non-hub vertex sets $V_{\overline{h}}$. (2) Secondly, we determine the edge connection between pairs of non-hub vertex sets. On two non-hub vertex sets $V_{\overline{h}}, V_{\overline{h}'}$, we connect two vertices $\mathbf{c}_{i,\overline{h}} \in V_{\overline{h}}$ and $\mathbf{c}_{i',\overline{h}'} \in V_{\overline{h}'}$, if $\mathbf{c}_{i,\overline{h}}$ and $\mathbf{c}_{i',\overline{h}'}$ are connected to the same vertex $\mathbf{c}_{k,h}$ on V_h.

After the two steps, every hub vertex $\mathbf{c}_{k,h}$, with $k \in \{1, .., K\}$ and all the non-hub vertices connected to $\mathbf{c}_{k,h}$ form the k-th clique Z_k. Therefore, the N-partite graph G is partitioned into K complete and disjoint subgraphs.

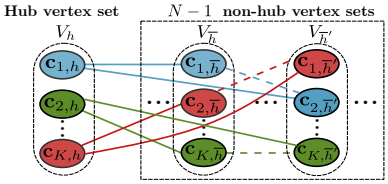


Figure 3: In the h-th iteration, $h \in \{1, ..., N\}$, the hub vertex set V_h is chosen and an assignment is computed by assigning the vertices between V_h and the $N-1$ non-hub vertex sets. Solid lines denote bipartite matching results between V_h and non-hub vertex sets (step (1)). Dashed lines denote connections between non-hub vertex sets (step (2)).

By iterating over all possible initial hub vertex sets $h \in \{1, ..., N\}$, we choose the assignment solution $f_{\hat{h}}$ which minimizes the assignment cost

$$f_{\hat{h}} = \underset{h \in \{1,...,N\}}{\arg\min} \sum_{(\mathbf{c},\mathbf{c}') \in E} f_h(\mathbf{c}, \mathbf{c}') \cdot w(\mathbf{c}, \mathbf{c}'), \quad (4)$$

where $f_h(\mathbf{c}, \mathbf{c}'), \forall (\mathbf{c}, \mathbf{c}') \in E$ is a binary indicator function that describes the edge connection: $f_h(\mathbf{c}, \mathbf{c}')$ equals 1 when two vertices $\mathbf{c}, \mathbf{c}'$ are connected. The assignment solution $f_{\hat{h}}$ describes the partition which leads to the K global clusters.

3.3. Frame Labeling by Viterbi Decoding

Given the embedded feature sequence $\mathbf{e}_{1\sim T_n,n}$ of video n, we determine the optimal label sequence $\hat{y}_{1\sim T_n,n}$. The posterior probability can be factorized into the product of likelihoods and the probability of a given temporal order, *i.e.*, $\hat{y}_{1\sim T_n,n} = \underset{y_{1\sim T_n,n}}{\arg\max}\, p(y_{1\sim T_n,n} | \mathbf{e}_{1\sim T_n,n}) = \underset{y_{1\sim T_n,n}}{\arg\max} \{\Pi_{i=1}^{T_n} p_n(\mathbf{e}_{i,n} | y_{i,n}) \cdot \Pi_{i=1}^{T_n} p_n(y_{i,n} | y_{1\sim i-1,n})\}$. We fit a Gaussian model on each global cluster and compute the frame-wise likelihoods, *i.e.*, $p_n(\mathbf{x}|k) = \mathcal{N}_k(\mathbf{x}; \mu_k, \Sigma_k), k \in \{1, ..., K\}$. The temporal order constraint is used to limit the search space for the optimal label sequence by filtering out the sequences that do not follow the temporal order.

The related works [21, 22] apply K-means on the frames of all the videos. From the unified clustering they derive only a single temporal order of clusters for all the videos. However, this is an unrealistic assumption due to interchangeable steps in the activities, *e.g.*, *pour milk* and *pour sugar* in *making coffee*. Instead, we can easily derive the temporal order for each video separately. We do so by sorting the within-video clusters according to the average timestamp of frames in each cluster. The output of the Viterbi decoding is the optimal cluster label sequence $\hat{y}_{1\sim T_n,n}$. More details are given in the supplementary.

4. Experiments

4.1. Datasets & Evaluation Metrics

We evaluate on Breakfast [42], the YouTube Instructions dataset (YTI) [36] and 50 Salads [43]. Breakfast is comprised of 1712 videos recorded in various kitchens. There are 10 composite activities of breakfast preparation. YTI is composed of 150 videos of 5 activities collected from YouTube. 50 Salads contains 50 videos of people preparing salads. Following [20, 21, 22], we use the dense trajectory Fisher vector features (DTFV) [44] for Breakfast and 50 Salads, and features provided by Alayrac *et al.* [36] on YTI. We use the evaluation protocol in [21] and report the performance in three metrics: (1) Mean over Frames (MoF) is the frame-level accuracy over the frames of all the videos. More frequent or longer action instances have a higher impact on the result. (2) Class-wise mean Intersection over Union (cIoU) is the average over the IoU performance for each class and penalizes segmentation results with dominating segments. (3) The F1-score penalizes results with oversegmentation.

4.2. Implementation Details

For our SSTEN, we adapt the number of dilated residual layers Q according to the dataset size: We set $Q = 5$ for YTI (15k frames per activity subset on average) and $Q = 10$ for Breakfast (360k) and 50 Salads (577k). The dimension of the hidden representation is set to 32. We set λ in Eq. (1) to 0.002 (Breakfast), 0.01 (YTI) and 0.005 (50 Salads). For clustering, we follow the protocol of [20, 36] and define the number of clusters K separately for each activity as the maximum number of ground truth classes. The values of K for the three datasets are provided in the supplementary material.

4.3. Comparison with the State-of-the-Art

We compare with unsupervised learning methods, as well as weakly and fully supervised approaches on Breakfast (Table 1), YTI (Table 2) and 50 Salads (Table 3). Most unsupervised segmentation approaches yield cluster-aware segments that are aligned across all the videos [20, 21, 22, 36, 45]. These approaches are evaluated with the **global Hungarian matching on all videos**, where the mapping between ground truth classes and clusters is performed on all the videos of an activity, which results in one mapping for each activity. The number of clusters K is set to the maximum number of ground truth classes for each activity (*i.e.*, K=**max.#gt**). We focus on the performance comparison in this setting and follow this setting in all the ablation studies.

Two recent approaches perform clustering (*i.e.*, TWFINTCH [37]) or category-agnostic boundary detection (*i.e.*, LSTM+AL [38]) on each video individually, without solving the alignment among different clusters or segments across videos. For a fair comparison, these are evaluated by **local Hungarian matching on individual videos**, where a per-video best ground-truth-to-cluster-label mapping is determined using the ground truth on each video separately. This results in a separate label mapping for each video. Following [37], we also report results with K set to the average number of actions for each activity (*i.e.*, K=**avg.#gt**) for a complete comparison.

In Table 1, TAEC achieves strong results in comparison to the unsupervised state-of-the-art and is even comparable to weakly supervised approaches. Although approaches without solving the alignment of clusters across videos inherently lead to better scores in the evaluation settings of the local Hungarian matching, our approach still compares favorably.

We compare qualitative results (with global Hungarian matching) of TAEC and MLP+kmeans [21] on 3 Breakfast activities in Fig. 4. We see that our two-step clustering (the 2nd rows in all *clustering result* plots) already leads to temporally consistent segments with relatively accurate boundaries of action instances, while K-means (the 4th rows in all *clustering result* plots) results in serious oversegmentation. The Viterbi decoding further improves the segmentation by suppressing the oversegmentation and domination of incorrect clusters (the 2nd rows in all *final result* plots). Moreover, MLP+kmeans [21] follows the constraint of the fixed temporal order of segments on videos of each activity (the 4th rows in all *final result* plots). In contrast, TAEC yields an individual temporal order for each video (the 2nd rows in all *final result* plots). Additional qualitative results and evaluation scores are included in the supplemental material.

For the YouTube Instructions dataset, we follow the protocol of [20, 21, 36] and report the results with and without considering background frames. Here, our TAEC outperforms all recent works in almost all of the metrics under all three settings.

50 Salads is a particularly challenging dataset for unsupervised approaches, as each video has a different order of actions and additionally includes many repetitive action instances. In the *eval*-level of 12 classes, TAEC outperforms all approaches under the global Hungarian matching evaluation and achieves competitive results under the local Hungarian matching. In the challenging *mid*-level evaluation of 19 classes, the sequential nature of frames is less advantageous. Therefore, MLP+kmeans [21] outperforms TAEC. Generally, in the local matching case, approaches without alignment across videos compare favorably.

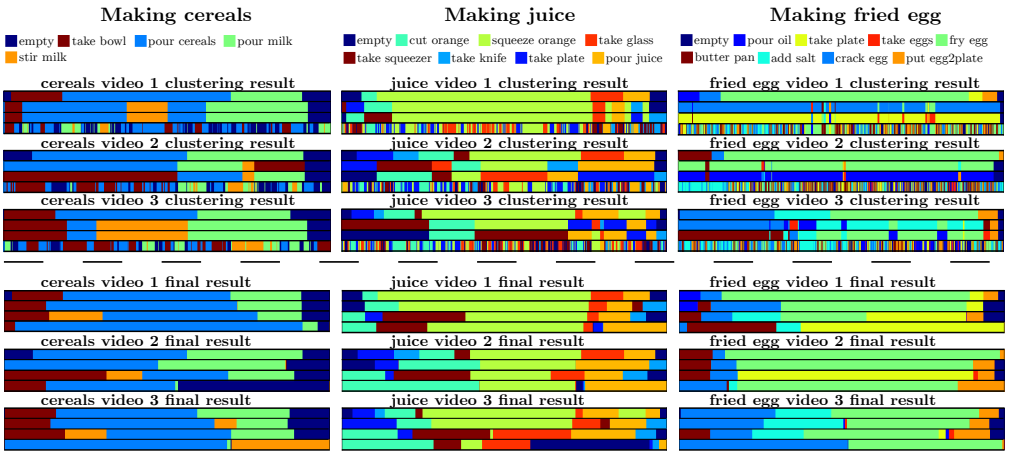


Figure 4: Qualitative results of clustering and final segmentation (with global Hungarian matching) on 3 activities (3 videos each) on Breakfast. For each video, the 4-row-group displays the ground truth (1st row), TAEC (2nd row), TAEC with naïve assignment (3rd row, quantitative comparison in Sec. 4.6), MLP+kmeans [21] (4th row). More quantitative and qualitative segmentation results are given in the supplementary.

Table 1
Comparison with state-of-the-art approaches on Breakfast (in %). * denotes approach without segment alignment across videos, ‡ denotes our reimplementation, underlined scores are acquired from the author.

Breakfast				
Approach	Supervision	MoF	IoU	F1
MSTCN++ [8]	full	67.6	-	-
G-FRNet [46]	full	67.7	-	-
DTGRM [10]	full	68.3	-	-
SSTDA [47]	full	70.3	-	-
BCN [9]	full	70.4	-	-
Global2local [48]	full	70.7	-	-
ASFormer [49]	full	**73.5**	-	-
NN-Viterbi [15]	weak	43.0	-	-
D3TW [12]	weak	45.7	-	-
TASL [50]	weak	47.8	-	-
CDFL [13]	weak	**50.2**	-	-
Global Hungarian matching on all videos (K = max. #gt)				
Mallow [20]	w/o	34.6	-	-
UNet+MLP [22]	w/o	48.1	-	29.9
ASAL [45]	w/o	**52.5**	-	**37.9**
MLP+kmeans [21]	w/o	41.8	-	26.4
MLP+kmeans ‡	w/o	42.9	13.1	25.5
TAEC	w/o	50.3	**19.0**	33.6
Local Hungarian matching on each video (K = **max.** #gt)				
LSTM+AL [38]*	w/o	42.9*	**46.9***	-
UNet+MLP [22]	w/o	52.2	-	-
TWFINTCH [37]*	w/o	57.8*	-	-
MLP+kmeans ‡	w/o	61.2	30.3	35.9
TAEC	w/o	**64.3**	41.2	**42.6**
Local Hungarian matching on each video (K = **avg.** #gt)				
TWFINCH [37]*	w/o	**62.7***	**42.3***	-
MLP+kmeans ‡	w/o	60.6	27.8	46.4
TAEC	w/o	62.6	32.3	**49.6**

Table 2
Comparison with the state-of-the-art on YTI (in %). * denotes approach without segment alignment across videos, ‡ denotes our reimplementation, underlined scores are acquired from the author.

YouTube Instructions						
Approach	Super-vision	MoF w/o bg	IoU w/o bg	F1 w/o bkg	MoF w bg	IoU w bg
Global Hungarian matching on all videos (K = max. #gt)						
Frank-Wolfe [36]	w/o	-	-	24.4	-	-
Mallow [20]	w/o	27.8	-	27.0	-	-
UNet+MLP [22]	w/o	28.9	8.3	29.9	-	-
ASAL [45]	w/o	44.9	-	**32.1**	-	-
MLP+kmeans [21]	w/o	39.0	9.8	28.3	14.5	9.6
MLP+kmeans ‡	w/o	39.4	9.9	29.6	14.4	9.7
TAEC	w/o	**46.6**	**10.7**	29.5	**17.0**	**10.5**
Local Hungarian matching on each video (K = **max.** #gt)						
LSTM+AL [38]*	w/o	-	-	39.7*	-	-
MLP+kmeans ‡	w/o	62.2	21.6	47.0	22.7	21.6
TAEC	w/o	**67.9**	**23.7**	**49.4**	**24.8**	**23.6**
Local Hungarian matching on each video (K = **avg.** #gt)						
TWFINCH [37]*	w/o	56.7*	-	48.2*	-	-
MLP+kmeans ‡	w/o	63.6	20.5	**52.3**	23.2	20.5
TAEC	w/o	**65.3**	**20.9**	51.0	**23.9**	**20.8**

4.4. Embedding and Clustering

We first evaluate our SSTEN embedding in combination with K-means and two-step clustering on three feature types: the AlexNet fc6 features [57] pre-trained on ImageNet [58], I3D features [59] pre-trained on the Kinetics dataset [60], and the precomputed dense trajectory Fisher vectors (DTFV) [44]. We also report results of raw features without any temporal embedding. For a fair comparison, we reduce the dimensions of the three

Table 3
Comparison with the state-of-the-art on 50 Salads (in %). * denotes approach without segment alignment across videos, ‡ denotes our reimplementation.

50 Salads					
Level	Approach	Supervision	MoF	IoU	F1
eval	STCNN [51]	full	72.0	-	-
	EDTCN [52]	full	73.4	-	-
	MA [53]	full	**88.5**	-	-
	Global Hungarian matching on all videos (K = max. #gt)				
	UNet+MLP [22]	w/o	30.6	-	-
	ASAL [45]	w/o	39.2	-	-
	MLP+kmeans [21]	w/o	35.5	-	-
	MLP+kmeans ‡	w/o	37.9	24.6	40.2
	TAEC	w/o	**48.4**	**26.0**	**44.8**
	Local Hungarian matching on each video (K = **max.** #gt)				
	LSTM+AL [38]*	w/o	**60.6***	-	-
	MLP+kmeans ‡	w/o	58.0	33.7	49.8
	TAEC	w/o	59.7	**35.0**	**54.4**
	Local Hungarian matching on each video (K = **avg.** #gt)				
	TWFINCH [37]*	w/o	**71.1***	-	-
	MLP+kmeans ‡	w/o	51.5	22.3	43.4
	TAEC	w/o	59.7	**35.7**	**54.7**
mid	SSTDA [47]	full	83.2	-	-
	MSTCN++ [8]	full	83.7	-	-
	HASR [54]	full	83.9	-	-
	ASRF [55]	full	84.5	-	-
	ASFormer [49]	full	**85.6**	-	-
	NNViterbi [15]	weak	49.4	-	-
	CDFL [56]	weak	**54.7**	-	-
	Global Hungarian matching on all videos (K = max. #gt)				
	UNet+MLP [22]	w/o	24.2	-	-
	ASAL [45]	w/o	**34.4**	-	-
	MLP+kmeans [21]	w/o	30.2	-	-
	MLP+kmeans ‡	w/o	29.1	**15.7**	**23.4**
	TAEC	w/o	26.6	14.9	**23.4**
	Local Hungarian matching on each video (K = **max.** #gt)				
	MLP+kmeans ‡	w/o	**55.6**	**29.6**	39.6
	TAEC	w/o	50.2	29.4	**40.3**
	Local Hungarian matching on each video (K = **avg.** #gt)				
	TWFINCH [37]*	w/o	**66.5***	-	-
	MLP+kmeans ‡	w/o	53.4	28.1	39.0
	TAEC	w/o	51.9	**30.2**	**41.9**

features without embedding to 32 via PCA. We conduct the experiments on Breakfast and report the results with and without our SSTEN embedding in Table 4.

Table 4
Comparison of features of SSTEN embedding, together with clustering methods on Breakfast (in %).

Model		K-means			Two-step clustering		
Feature	Embedding	MoF	IoU	F1	MoF	IoU	F1
AlexNet	w/o	25.9	11.3	22.3	33.7	10.7	20.2
I3D	w/o	**33.4**	**14.4**	**25.6**	**37.7**	**14.3**	**26.1**
DTFV	w/o	30.8	11.8	23.0	34.5	12.1	22.0
AlexNet	SSTEN	33.0	14.2	27.0	39.1	16.1	30.7
I3D	SSTEN	37.9	**18.7**	**33.3**	45.2	**20.5**	**35.1**
DTFV	SSTEN	**39.3**	17.8	31.9	**50.3**	19.0	33.6

Comparison of raw features without embedding. Among the three types of features without temporal embedding, I3D achieves the best performance, while AlexNet features lead to the worst results. AlexNet features are computed from individual spatial frames. On the contrary, each frame feature of DTFV and I3D is computed based on a chunk of its temporal neighbor frames. Therefore, the features already carry intrinsic temporal consistency. Furthermore, the two-stream I3D model can leverage both RGB and optical flow. Therefore, I3D features achieve a better performance than DTFV, which rely on handcrafted dense trajectories.

Comparison of SSTEN embeddings learned on different features. When comparing the SSTEN embeddings to the performance of the raw features, we see that SSTEN leads to a significant performance gain for both clustering methods. For DTFV, the performance improvements by SSTEN are MoF 8.5%, IoU 6.0%, F1 8.9% with K-means and MoF 15.8%, IoU 6.9%, F1 11.6% with two-step clustering.

Among the three types of SSTEN embedded features, I3D has slightly better IoU and F1 scores while DTFV leads to the best MoF scores for both, K-means and the two-step clustering. Overall, the SSTEN embeddings learned from these two features perform comparably. We conduct the following experiments using DTFV, which is also used in related works.

4.5. Impact of Loss Terms on Clustering

To evaluate the impact of the two loss terms in Eq. (1), we plot the quantitative segmentation results of SSTEN with both K-means and the two-step clustering w.r.t. different reconstruction loss coefficients λ in Fig. 5. In general, two-step clustering leads to a better performance than K-means for almost all λ values (except for the case of only reconstruction loss). With decreasing λ, the relative time prediction loss has an increasing impact and the embedded features have better global temporal consistency, which explains the increasing IoU and F1 scores. However, at extremely small λ values, the embedded features overfit to the relative time prediction task, which results in saturated IoU and F1 scores, and a significant drop in MoF for both K-means and two-step clustering.

To intuitively illustrate the loss term impact on the two-step clustering, we plot the similarity matrices for SSTEN embeddings trained with three different λ in Fig. 6. Here, we look at the similarity matrices with temporal Gaussian kernel (bottom row). Intuitively, the similarity matrix with clear diagonal block structure (Fig. 6(a2)), which is the result of an appropriate ratio between the reconstruction loss and relative time prediction loss ($\lambda = 0.002$), leads to the best segmentation performance. When λ becomes larger (*e.g.*, $\lambda = 0.01$), the reconstruction loss has a larger impact and the diagonal block structure

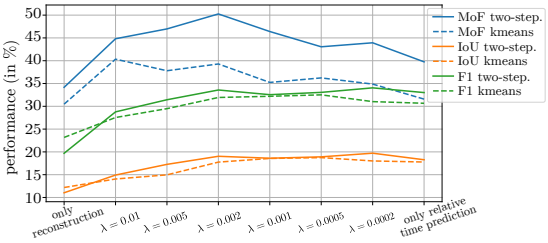


Figure 5: Segmentation performance of both clustering methods on SSTEN embeddings with different λ on Breakfast.

(Fig. 6(b2)) becomes pale. Therefore, the performances of embedded features with $\lambda = 0.005$, $\lambda = 0.01$ and only reconstruction loss degrade successively. On the other hand, for extremely small λ values (*e.g.*, $\lambda = 0.0005$), the block diagonal structure (Fig. 6(c2)) becomes noisy due to overfitting on relative time prediction.

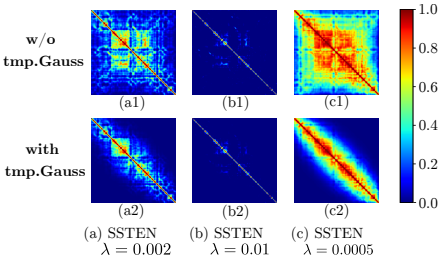


Figure 6: Frame-to-frame similarity matrices of SSTEN embeddings for the same Breakfast video. Columns show the similarity matrices for different λ, while the rows show results without (top) and with (bottom) temporal Gaussian kernel.

Therefore, both the reconstruction and the relative timestamp prediction loss, when combined in an appropriate ratio, are indispensable to learn the effective representation that preserves both spatial layout and the temporal information.

4.6. Impact of Cluster Assignment

In this ablation study, we evaluate the efficacy of the global cluster assignment. For two-step clustering, we evaluate two strategies of grouping within-video clusters into global clusters: (1) the naïve assignment, for which we order the sub-clusters according to the average timestamp and simply group the k-th sub-clusters of all videos into a global cluster, *i.e.*, the global cluster $Z_k = \{\mathbf{c}_{k,n} | n = 1, .., N\}$, and (2) the global cluster assignment, as detailed in Sec. 3.2.2.

In order to show how the different cluster assignment strategies affect the clustering result, we report both, the results of the two-step clustering (before Viterbi decoding) and the final segmentation performance (after Viterbi decoding) on Breakfast and 50 Salads in Table 5. The global cluster assignment outperforms the naïve assignment by a large margin for both, clustering results and the final segmentation results, on both datasets. The advantage of the global cluster assignment is even more evident on 50 Salads.

Table 5
Impact of cluster assignment strategies for two-step clustering on SSTEN embeddings on Breakfast and 50 Salads (*eval* level, *i.e.*, 12 classes) (in %).

Dataset	Strategy	Clustering			Final		
		MoF	IoU	F1	MoF	IoU	F1
Breakfast	global cluster	**38.6**	**13.7**	**25.9**	**50.3**	**19.0**	**33.6**
	naïve	25.1	12.4	23.9	42.3	17.7	32.0
50 Salads	global cluster	**45.3**	**23.0**	**43.7**	**48.4**	**26.0**	**44.8**
	naïve	28.9	16.2	32.9	35.0	22.7	38.0

We illustrate exemplary qualitative results of the clustering and the final segmentation for 3 activities (with 3 videos each) on Breakfast in Fig. 4. For each video, the 4-row group displays the ground truth (1st row), the result with global cluster assignment (2nd row) and the result with naïve assignment (3rd row). The 4th row shows the result of MLP+kmeans [21]. By comparing all the 3rd rows of *cereals video [id] final result* in Fig. 4, we see that the naïve assignment simply assumes the sub-clusters in the same temporal order in each video belong to the same global cluster, while they might not be close to each other in the feature space. On the contrary, the global cluster assignment (the 2nd rows of *cereals video [id] final result*) yields an optimal assignment solution with respect to the pairwise distances between sub-clusters, resulting in different orderings of sub-clusters on each video. Note that on some videos, global cluster assignment could lead to the same assignment result as naïve assignment.

5. Conclusion

We proposed a new pipeline for the unsupervised learning of action segmentation. For the feature embedding, we propose a temporal-aware embedding network that performs sequence-to-sequence learning with the pretext tasks of relative timestamp prediction and feature reconstruction. For clustering, we propose a two-step clustering schema, consisting of within-video clustering and cross-video global cluster assignment. The temporal embedding of sequence-to-sequence learning together with two-step clustering is proven to be a well-suitable combination that considers the sequential nature of frames in both processing steps. Ultimately, we combine the temporal embedding with a frame-to-cluster assignment based on Viterbi decoding, which achieves the unsupervised state-of-the-art on three challenging benchmarks.

References

[1] Z. Wang, Q. She, A. Smolic, Action-net: Multipath excitation for action recognition, in: CVPR, 2021.

[2] C. Feichtenhofer, X3d: Expanding architectures for efficient video recognition, in: CVPR, 2020.

[3] C. Yang, Y. Xu, J. Shi, B. Dai, B. Zhou, Temporal pyramid network for action recognition, in: CVPR, 2020.

[4] M. Xu, C. Zhao, D. S. Rojas, A. Thabet, B. Ghanem, G-tad: Sub-graph localization for temporal action detection, in: CVPR, 2020.

[5] P. Zhao, L. Xie, C. Ju, Y. Zhang, Y. Wang, Q. Tian, Bottom-up temporal action localization with mutual regularization, in: ECCV, 2020.

[6] Y. Bai, Y. Wang, Y. Tong, Y. Yang, Q. Liu, J. Liu, Boundary content graph neural network for temporal action proposal generation, in: ECCV, 2020.

[7] Y. A. Farha, J. Gall, MS-TCN: Multi-stage temporal convolutional network for action segmentation, in: CVPR, 2019.

[8] S. Li, Y. A. Farha, Y. Liu, M.-M. Cheng, J. Gall, MS-TCN++: Multi-Stage Temporal Convolutional Network for Action Segmentation, PAMI (2020).

[9] Z. Wang, Z. Gao, L. Wang, Z. Li, G. Wu, Boundary-aware cascade networks for temporal action segmentation, in: ECCV, 2020.

[10] D. Wang, D. Hu, X. Li, D. Dou, Temporal relational modeling with self-supervision for action segmentation, in: AAAI, 2021.

[11] Y. Huang, Y. Sugano, Y. Sato, Improving action segmentation via graph-based temporal reasoning, in: CVPR, 2020.

[12] C.-Y. Chang, D.-A. Huang, Y. Sui, L. Fei-Fei, J. C. Niebles, D3tw: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation, in: CVPR, 2019.

[13] J. Li, P. Lei, S. Todorovic, Weakly supervised energy-based learning for action segmentation, in: ICCV, 2019.

[14] A. Richard, H. Kuehne, J. Gall, Weakly supervised action learning with RNN based fine-to-coarse modeling, in: CVPR, 2017.

[15] A. Richard, H. Kuehne, A. Iqbal, J. Gall, NeuralNetwork-Viterbi: A framework for weakly supervised video learning, in: CVPR, 2018.

[16] D.-A. Huang, F.-F. Li, J. C. Niebles, Connectionist temporal modeling for weakly supervised action labeling, in: ECCV, 2016.

[17] M. Fayyaz, J. Gall, SCT: Set constrained temporal transformer for set supervised action segmentation, in: CVPR, 2020.

[18] J. Li, S. Todorovic, Set-constrained Viterbi for set-supervised action segmentation, in: CVPR, 2020.

[19] A. Richard, H. Kuehne, J. Gall, Action sets: Weakly supervised action segmentation without ordering constraints, in: CVPR, 2018.

[20] F. Sener, A. Yao, Unsupervised learning and segmentation of complex activities from video, in: CVPR, 2018.

[21] A. Kukleva, H. Kuehne, F. Sener, J. Gall, Unsupervised learning of action classes with continuous temporal embedding, in: CVPR, 2019.

[22] R. G. VidalMata, W. J. Scheirer, H. Kuehne, Joint visual-temporal embedding for unsupervised learning of actions in untrimmed sequences, in: WACV, 2021.

[23] B. L. Bhatnagar, S. Singh, C. Arora, C. V. Jawahar, Unsupervised learning of deep feature representation for clustering egocentric actions, in: IJCAI, 2017.

[24] N. Srivastava, E. Mansimov, R. Salakhudinov, Unsupervised learning of video representations using LSTMs, in: ICML, 2015.

[25] E. L. Denton, V. Birodkar, Unsupervised learning of disentangled representations from video, in: NeurIPS, 2017.

[26] R. Villegas, J. Yang, S. Hong, X. Lin, H. Lee, Decomposing motion and content for natural video sequence prediction, in: ICLR, 2017.

[27] D. Kim, D. Cho, I. S. Kweon, Self-supervised video representation learning with space-time cubic puzzles, in: AAAI, 2019.

[28] H.-Y. Lee, J.-B. Huang, M. Singh, M.-H. Yang, Unsupervised representation learning by sorting sequences, in: ICCV, 2017.

[29] V. Ramanathan, K. Tang, G. Mori, L. Fei-Fei, Learning temporal embeddings for complex video analysis, in: ICCV, 2015.

[30] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, T. Tuytelaars, Modeling video evolution for action recognition, in: CVPR, 2015.

[31] A. Cherian, B. Fernando, M. Harandi, S. Gould, Generalized rank pooling for activity recognition, in: CVPR, 2017.

[32] M. Hoai, F. D. la Torre, Maximum margin temporal clustering, in: Artificial Intelligence and Statistics, 2012.

[33] S. Li, K. Li, Y. Fu, Temporal subspace clustering for human motion segmentation, in: CVPR, 2015.

[34] S. Tierney, J. Gao, Y. Guo, Subspace clustering for sequential data, in: CVPR, 2014.

[35] Y. Zhang, S. Tang, H. Sun, H. Neumann, Human motion parsing by hierarchical dynamic clustering, in: BMVC, 2018.

[36] J.-B. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, S. Lacoste-Julien, Unsupervised learning from narrated instruction videos, in: CVPR, 2016.

[37] M. S. Sarfraz, N. Murray, V. Sharma, A. Diba, L. Van Gool, R. Stiefelhagen, Temporally-weighted

hierarchical clustering for unsupervised action segmentation, in: CVPR, 2021.
[38] S. N. Aakur, S. Sarkar, A perceptual prediction framework for self supervised event segmentation, in: CVPR, 2019.
[39] L. Zelnik-Manor, P. Perona, Self-tuning spectral clustering, in: NeurIPS, 2005.
[40] W. P. Pierskalla, Letter to the editor—the multidimensional assignment problem, Operations Research 16 (1968) 422–431.
[41] H.-J. Bandelt, Y. Crama, F. C. Spieksma, Approximation algorithms for multi-dimensional assignment problems with decomposable costs, Discrete Applied Mathematics 49 (1994) 25–50.
[42] H. Kuehne, A. Arslan, T. Serre, The language of actions: Recovering the syntax and semantics of goal-directed human activities, in: CVPR, 2014.
[43] S. Stein, S. J. McKenna, Combining embedded accelerometers with computer vision for recognizing food preparation activities, in: ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2013.
[44] H. Wang, C. Schmid, Action recognition with improved trajectories, in: ICCV, 2013.
[45] J. Li, S. Todorovic, Action shuffle alternating learning for unsupervised action segmentation, in: CVPR, 2021.
[46] D. Wang, Y. Yuan, Q. Wang, Gated forward refinement network for action segmentation, Neurocomputing 407 (2020) 63–71.
[47] M.-H. Chen, B. Li, Y. Bao, G. AlRegib, Z. Kira, Action segmentation with joint self-supervised temporal domain adaptation, in: CVPR, 2020.
[48] S.-H. Gao, Q. Han, Z.-Y. Li, P. Peng, L. Wang, M.-M. Cheng, Global2local: Efficient structure search for video action segmentation, in: CVPR, 2021.
[49] F. Yi, H. Wen, T. Jiang, Asformer: Transformer for action segmentation, in: BMVC, 2021.
[50] Z. Lu, E. Elhamifar, Weakly-supervised action segmentation and alignment via transcript-aware union-of-subspaces learning, in: ICCV, 2021.
[51] C. Lea, A. Reiter, R. Vidal, G. D. Hager, Segmental spatiotemporal cnns for fine-grained action segmentation, in: ECCV, 2016.
[52] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, G. D. Hager, Temporal convolutional networks for action segmentation and detection, in: CVPR, 2017.
[53] C. Fermüller, F. Wang, Y. Yang, K. Zampogiannis, Y. Zhang, F. Barranco, M. Pfeiffer, Prediction of manipulation actions, International Journal of Computer Vision 126 (2018) 358–374.
[54] H. Ahn, D. Lee, Refining action segmentation with hierarchical video representations, in: ICCV, 2021.
[55] Y. Ishikawa, S. Kasai, Y. Aoki, H. Kataoka, Alleviating over-segmentation errors by detecting action boundaries, in: WACV, 2021.
[56] J. Liu, A. Shahroudy, M. L. Perez, G. Wang, L.-Y. Duan, A. K. Chichung, NTU RGB+D 120: A large-scale benchmark for 3D human activity understanding, PAMI (2019).
[57] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: NeurIPS, 2012.
[58] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: CVPR, 2009.
[59] J. Carreira, A. Zisserman, Quo vadis, action recognition? A new model and the kinetics dataset, in: CVPR, 2017.
[60] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al., The kinetics human action video dataset, arXiv preprint arXiv:1705.06950 (2017).

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.*,

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

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

CEUR Workshop Proceedings (CEUR-WS.org)

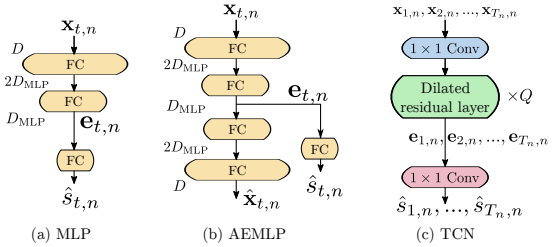


Figure 1: Three baseline variants of embedding networks.

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.*,

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

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.

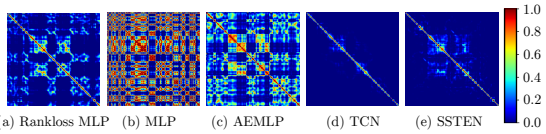
MLP uses three FC layers for relative timestamp prediction. *AEMLP* uses MLP-based autencoder for both relative timestamp prediction and feature reconstruction. *TCN* deploys Q stacked dilated residual layers only for relative timestamp prediction.

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.

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
DTFV	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
	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.

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.

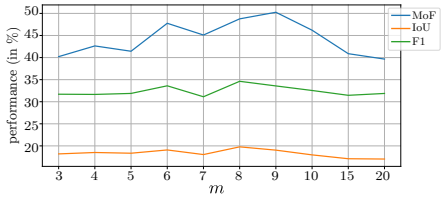


Figure 3: Segmentation performance of the two-step clustering on SSTEN embeddings ($\lambda = 0.002$) with different m (for local scaling) on Breakfast.

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.

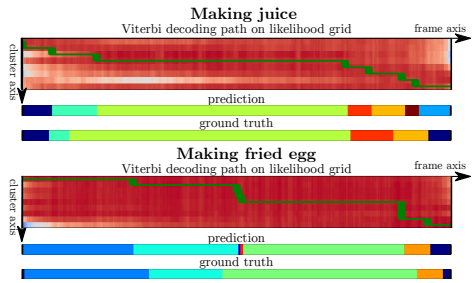


Figure 4: Comparison of Viterbi decoding path on the likelihood grid computed from the global clusters resulted from two-step clustering on the SSTEN embeddings, for two videos on Breakfast. The warm (red)/cool (blue) colors in the grid indicate high/low likelihoods of a frame belonging to an action 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.

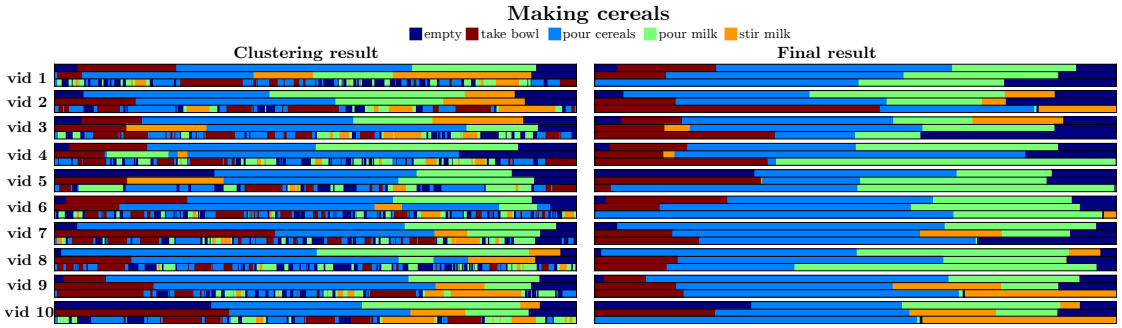


Figure 5: Qualitative results of clustering and final segmentation of 10 *making cereals* videos on Breakfast. For each video, the 3-row-group displays the ground truth (1st row), TAEC (2nd row), MLP+kmeans [12] (3rd row). The mapping between cluster labels and ground truth classes is done with global 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).

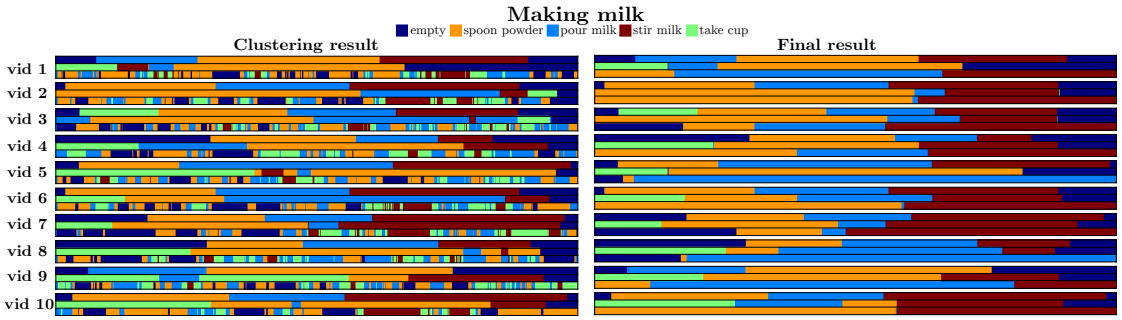


Figure 6: Qualitative results for 10 *making milk* videos on Breakfast.

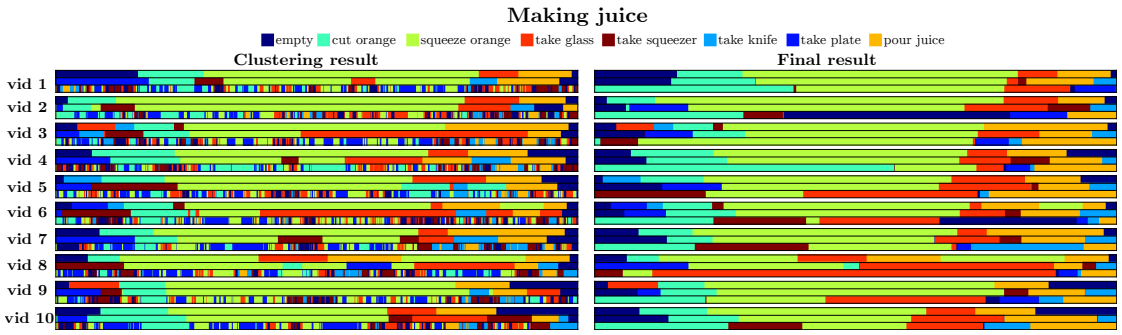


Figure 7: Qualitative results for 10 *making juice* videos on Breakfast.

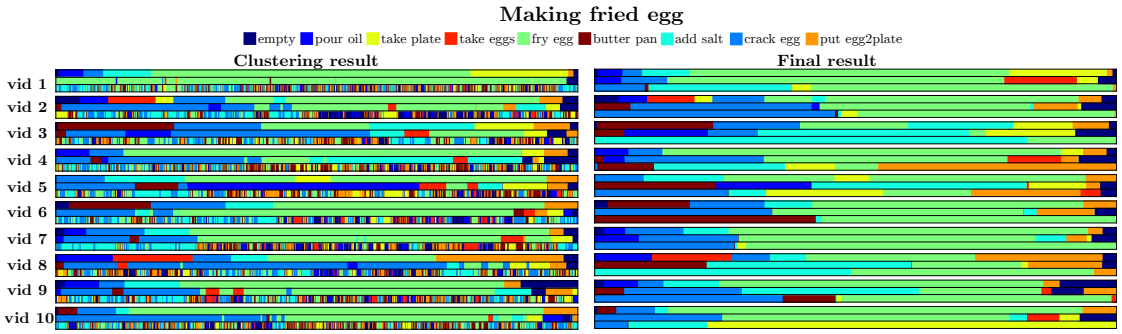


Figure 8: Qualitative results for 10 *making fried egg* videos on Breakfast.

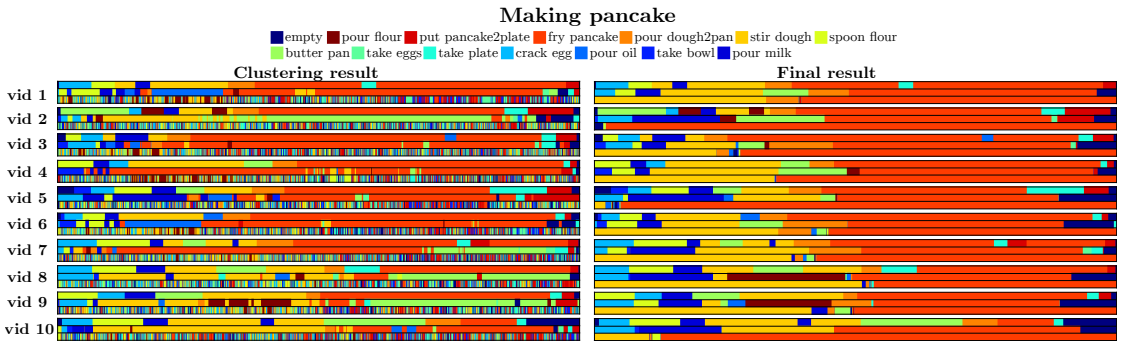


Figure 9: Qualitative results for 10 *making pancake* videos on Breakfast.

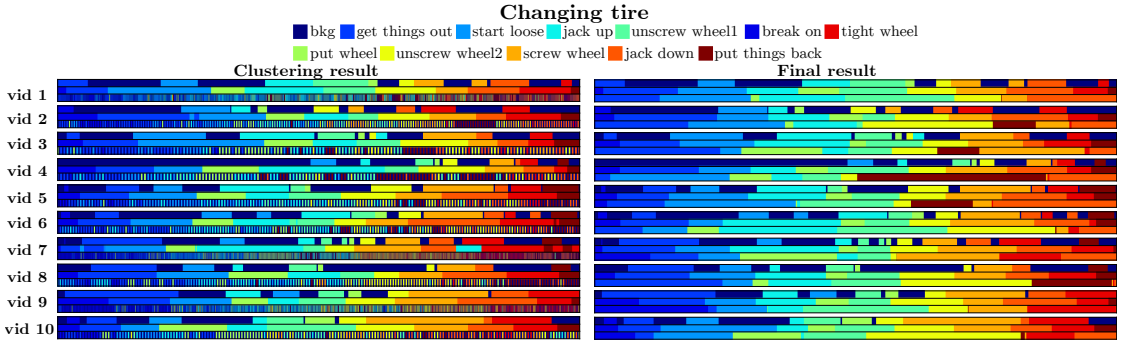


Figure 10: Qualitative results for 10 *changing tire* videos of the YouTube Instructions dataset. Similar to the Breakfast illustrations, for each video the 3-row-group shows the ground truth (1st row), TAEC (2nd row), and MLP+kmeans [12] (3rd row).

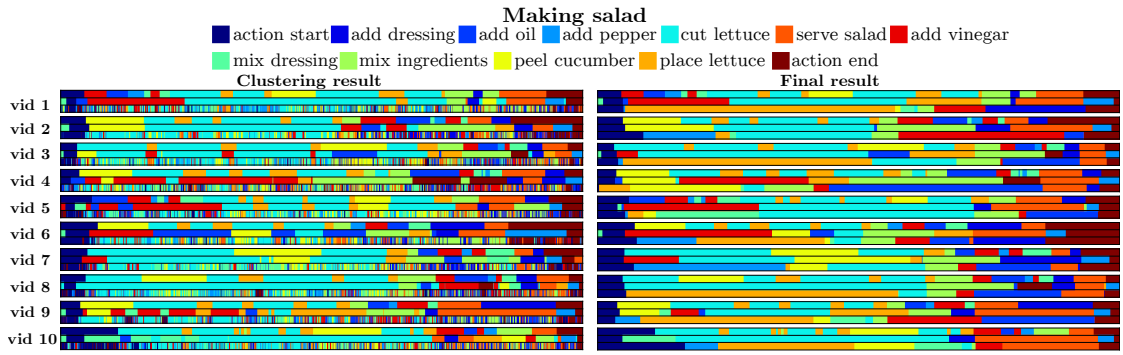


Figure 11: Qualitative results for 10 *making salad* videos on the 50 Salads dataset (from the *eval*-level, 12 action classes). Again, for each video the 3-row-group shows the ground truth (1st row), TAEC (2nd row), and MLP+kmeans [12] (3rd row).