Temporally-Weighted Hierarchical Clustering for Unsupervised Action Segmentation

M. Saquib Sarfraz[1,7], Naila Murray[2], Vivek Sharma[1,3,4], Ali Diba[5], Luc Van Gool[5,6], Rainer Stiefelhagen[1]

[1] Karlsruhe Institute of Technology, [2] Facebook AI Research,

[3] MIT, [4] Harvard Medical School, [5] KU Leuven, [6] ETH Zurich [7] Daimler TSS

Abstract

Action segmentation refers to inferring boundaries of semantically consistent visual concepts in videos and is an important requirement for many video understanding tasks. For this and other video understanding tasks, supervised approaches have achieved encouraging performance but require a high volume of detailed frame-level annotations. We present a fully automatic and unsupervised approach for segmenting actions in a video that does not require any training. Our proposal is an effective temporally-weighted hierarchical clustering algorithm that can group semantically consistent frames of the video. Our main finding is that representing a video with a 1-nearest neighbor graph by taking into account the time progression is sufficient to form semantically and temporally consistent clusters of frames where each cluster may represent some action in the video. Additionally, we establish strong unsupervised baselines for action segmentation and show significant performance improvements over published unsupervised methods on five challenging action segmentation datasets. Our code is available.[1]

1. Introduction

Human behaviour understanding in videos has traditionally been addressed by inferring high-level semantics such as activity recognition [12, 3]. Such works are often limited to tightly clipped video sequences to reduce the level of labelling ambiguity and thus make the problem more tractable. However, a more fine-grained understanding of video content, including for un-curated content that may be untrimmed and therefore contain a lot of material unrelated to human activities, would be beneficial for many downstream video understanding applications. Consequently, the less-constrained problem of action segmentation in untrimmed videos has received increasing attention. Action segmentation refers to labelling each frame of a video with an action, where the sequence of actions is usually performed by a human engaged in a high-level activity such as making coffee (illustrated in Figure 1). Action segmentation is more challenging than activity recognition of trimmed videos for several reasons, including the presence of background frames that don't depict actions of relevance to the high-level activity. A major challenge is the need for significantly more detailed annotations for supervising learning-based approaches. For this reason, weakly- and unsupervised approaches to action segmentation have gained popularity [34, 27, 18, 32]. Some approaches have relied on natural language text extracted from accompanying audio to provide frame-based action labels for training action segmentation models [2]. This of course makes the strong assumption that audio and video frames are well-aligned. Other approaches assume some *a priori* knowledge of the actions, such as the high-level activity label or the list of actions depicted, in each video [11, 34]. Even this level of annotation however, requires significant annotation effort for each training video as not all activities are performed using the same constituent actions.

Most weakly- and unsupervised methods, whatever their degree of *a priori* knowledge, focus on acquiring pseudo-labels that can be used to supervise training of task-specific feature embeddings [22, 32, 9, 26, 28, 34]. As pseudo-labels are often quite noisy, their use may hamper the efficacy of the learned embeddings. In this work, we adopt the view that action segmentation is fundamentally a *grouping* problem, and instead focus on developing clustering methods that effectively delineate the temporal boundaries between actions. This approach leads to an illuminating finding: for action segmentation, a simple clustering (*e.g.*, with Kmeans) of appearance-based frame features achieves performance on par with, and in some cases superior to, SoTA weakly-supervised and unsupervised methods that require training on the target video data (please refer to section 3 for details). This finding indicates that a sufficiently discriminative visual representation of video frames can be used to group frames into visually coherent clusters. However, for action segmentation, temporal coherence is also critically

[1] https://github.com/ssarfraz/FINCH-Clustering/tree/master/TW-FINCH

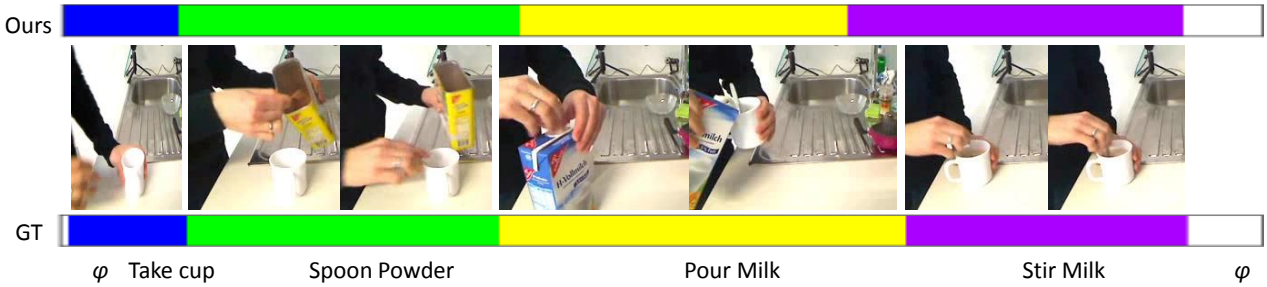


Figure 1. Segmentation output example from Breakfast Dataset [14]: *P46_webcam02_P46_milk*. Colors indicate different actions in chronological order: φ, take_cup, spoon_powder, pour_milk, stir_milk, φ, where φ is background shown in while color.

important. Building on these insights, we adapt a hierarchical graph-based clustering algorithm to the task of temporal video segmentation by modulating appearance-based graph edges between frames by their temporal distances. The resulting spatio-temporal graph captures both visually- and temporally-consistent neighbourhoods of frames that can be effectively extracted. Our work makes the following main contributions:

- We establish strong appearance-based clustering baselines for unsupervised action segmentation that outperform SoTA models;
- We propose to use temporally-modulated appearance-based graphs to represent untrimmed videos;
- We combine this representation with a hierarchical graph-based clustering algorithm in order to perform temporal action segmentation.

Our proposed method outperforms our strong baselines and existing SOTA unsupervised methods by a significant margin on 5 varied and challenging benchmark datasets.

2. Related Work

There exists a large body of work on spatial and spatio-temporal action recognition in videos (see [12, 3] for recent surveys). In this section we review works related to our problem of interest, temporal action segmentation, focusing on weakly- and unsupervised methods.

Most existing temporal action segmentation methods, be they fully supervised [8, 14, 17], weakly supervised [34, 22, 28, 11] or unsupervised [18, 32, 2], use frame-level annotations to train their models. They differ in whether the annotations are collected by human annotators or extracted in a semi- or unsupervised manner. These models largely follow a paradigm in which an embedding is trained on top of pre-extracted frame-level video features, such as I3D [5], as in [34, 11], or hand-crafted video features such as improved dense trajectories IDT [38], as in [22, 32, 18, 9, 26]. To train the embedding, a discriminative objective function is used in conjunction with the collected annotations [22, 32, 9, 26, 28, 34]. Weakly-supervised and unsupervised methods, discussed next, vary largely in the manner in which they extract and exploit pseudo-labels.

Weakly-supervised methods generally assume that both the video-level activity label and the ordering of actions, termed transcripts, are known during training. Some weakly-supervised works have a two-stage training process where pseudo-labels are first generated using transcripts and then used to train a frame classification network [16, 26]. In contrast, the method NN-Vit [28] directly leverages transcripts while learning a frame classification model. For this they introduce a loss based on Viterbi decoding that enforces consistency between frame-level label predictions. In a similar spirit, a recent proposal called MuCoN [34] aims to leverage transcripts while learning a frame classification model. They learn two network branches, only one of which has access to transcripts, while ensuring that both branches are mutually consistent. Another recent method called CDFL [22] also aims to use transcripts when training their frame labelling model. They first build a fully-connected, directed segmentation graph whose paths represent actions. They then train their model by maximizing the energy difference between valid paths ($i.e$ paths that are consistent with the ground-truth transcript) and invalid ones. In SCT [11], the authors assume that the set of action labels for a given video, but not their order, is known. They determine the ordering and temporal boundaries of the actions by alternatively optimizing set and frame classification objectives to ensure that frame-level action predictions are consistent with the set-level predictions.

Unsupervised methods generally assume knowledge only of the video-level activity label [32, 2, 18, 1, 36]. In Mallow [32], the authors use video-level annotations in an iterative approach to action segmentation, alternating optimization of a discriminative appearance model and a generative temporal model of action sequences. In Frank-Wolfe [2], video narrations are extracted using ASR and used to extract an action sequence for a set of videos of an activity. This is accomplished by separately clustering the videos and the ASR-recovered speech to identify action verbs in the spe-

cific video. Temporal localization is then obtained by training a linear classifier. CTE [18] proposes to learn frame embeddings that incorporate relative temporal information. They train a video activity model using pseudo-labels generated from Kmeans clustering of the videos' IDT features. The trained embeddings are then re-clustered at the ground-truth number of actions and ordered using statistics of the relative time-stamps with a GMM+Viterbi decoding. VTE-UNET [36] uses similarly learned embeddings in combination with temporal embeddings to improve upon [18]. Another interesting approach is LSTM+AL [1], which fine-tunes a pre-trained VGG16 model with an LSTM, using future frame prediction as a self-supervision objective, to learn frame embeddings. These embeddings are then used to train an action boundary detection model.

All of these methods require training on the target video dataset, which from a practical standpoint is a very restrictive requirement. In contrast, our method does not require any training, and relies only on frame clustering to segment a given video.

3. Method

As mentioned in the introduction, unsupervised temporal video segmentation is inherently a grouping and/or clustering problem. We observe that, given a relatively good video frame representation, the boundaries of actions in a video are discernible without the need for further training on objectives that use noisy pseudo-labels, something that almost all current methods pursue. To substantiate this observation and to have a basis for our later discussion we provide results of directly clustering a commonly used untrimmed video benchmark (Breakfast dataset [14] with 1712 videos) in Table 1. The goal of clustering is to group the frames of each video into its ground-truth actions. We consider two representative clustering methods: (1) Kmeans [23], representing centroid-based methods; and (2) a recent proposal called FINCH [31], representing state-of-the-art hierarchical agglomerative clustering methods. We cluster the extracted 64-dim IDT features of each video to its required number of actions (clusters). The performance is computed by mapping the estimated cluster labels of each video to the ground-truth labels using the Hungarian method, and the accuracy is reported as mean over frames (MoF). Section 4 contains more details about the experimental setup. As can be seen, simple clustering baselines Kmeans/FINCH performs at par with the best reported weakly/un-supervised methods in this video level evaluation. These results establish new, strong baselines for temporal video segmentation, and suggest that focusing on more specialized clustering techniques may be promising.

Among existing clustering methods, hierarchical clustering methods such as [31] are an attractive choice for the task at hand, as such methods provide a hierarchy of partitions of the data as opposed to a single partition. In this paper we adopt a hierarchical clustering approach to action segmentation that does not require video-level activity labels. In contrast, the existing body of work requires not only such prior knowledge but also requires training on the target video data. The ability to generate a plausible video segmentation without relying on training is highly desirable from a practical standpoint. To the best of our knowledge there is no existing prior work that addresses this challenging and practical scenario.

	Weakly Sup.	Unsupervised			
	CDFL [22]	LSTM+AL [1]	VTE-UNET [36]	Kmeans	FINCH
MoF	50.2	42.9	52.2	42.7	51.9

Table 1. Simple clustering with Kmeans or FINCH is competitive with the best reported weakly or unsupervised methods.

Our proposal is similar in spirit to the FINCH [31] algorithm. The authors in [31] make use of the observation that the nearest and the shared neighbor of each sample can form large linking chains in the data. They define an adjacency matrix that links all samples to their nearest first neighbour, thus building a 1-nearest neighbor (1-NN) graph. They showed that the connected components of this adjacency graph partitions the data into fixed clusters. A recursive application of this on the obtained partition(s) yields a hierarchy of partitions. The algorithm typically provides hierarchical partitions of the data in only a few recursion steps.

Based on this observation of finding linking chains in the data with nearest or shared neighbours, we propose a similar hierarchical clustering procedure for the problem of temporal video segmentation. We propose to use a spatio-temporal graphical video representation by linking frames based on their feature space proximity and their respective positions in time. In particular, we would like this representation to encode both feature-space and temporal proximity. We achieve this by using time progression as a modulating factor when constructing the graph.

For a video with N frames $X = \{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}\}$, we define a directed graph $G = (V, E)$ with edges describing the proximity of frames in feature space and time. We construct G by computing the frames' feature space distances and then modulating them by their respective temporal positions, using the following:

$$G_f(i,j) = \begin{cases} 1 - \langle \mathbf{x_i}, \mathbf{x_j} \rangle & \text{if } i \neq j \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

where G_f represents a graph with edge weights computed in the feature-space. The inner product is computed on L2-normalized feature vectors to ensure that the distance is in the $[0, 1]$ range. A similar graph G_t is defined in the time-space, and edge weights are computed from the time-stamps. For N frames the time-stamps are defined as

$T = \{1, 2, \cdots, N\}$, and the edge weights are computed as:

$$G_t(i,j) = \begin{cases} |t_i - t_j| / N & \text{if } i \neq j \\ 1 & \text{otherwise} \end{cases} \quad (2)$$

The edges in Equation 2 represent the temporal difference between the nodes, weighted by the total length of the sequence. Because we want to use the temporal graph as a modulating factor for the feature-space graph, The term $|t_i - t_j| / N$ provides a weighing mechanism relative to the sequence length. We then compute temporally-modulated appearance-based distances as follows:

$$W(i,j) = G_f(i,j) \cdot G_t(i,j). \quad (3)$$

$W(i,j)$ therefore specifies the temporally weighted distance between graph nodes ($i.e.$ frames) i and j. Finally, from this we construct a 1-NN graph by keeping only the closest node to each node (according to $W(i,j)$) and setting all other edges to zero.

$$G(i,j) = \begin{cases} 0 & \text{if } W(i,j) > \min\limits_{\forall j} W(i,j) \\ 1 & \text{otherwise} \end{cases} \quad (4)$$

The 1-NN temporal graph G defines an adjacency matrix where each node is linked to its closest neighbor according to the temporally weighted distances W. For all non-zero edges $G(i,j)$, we make the links symmetric by setting $G(j,i) = 1$. This results in a symmetric sparse matrix that encodes both feature space and temporal distances, and whose connected components form clusters. Note that Equation 4 only creates absolute links in the graph and we do not perform any additional graph segmentation steps. In contrast, popular methods that build similar nearest-neighbour graphs, such as spectral clustering [37], need to solve a graph-cut problem that involves solving an eigenvalue decomposition and thus have cubic complexities.

The connected components of the graph in Equation 4 automatically partition the data into discovered clusters. We use a recursive procedure to obtain further successive groupings of this partition. Each step forms groups of previously-obtained clusters, and the recursion terminates when only one cluster remains. Because in each recursion the graph's connected components form larger linking chains [31], in only a few recursions a small set of hierarchical partitions can be obtained, where each successive partition contains clusters of the previous partition's clusters.

The main steps of the proposed algorithm are shown in Algorithm 1. After computing the temporal 1-NN graph through Equations 1-4, its connected components provide the first partition. We then merge these clusters recursively based on the cluster averages of features and time-stamps.

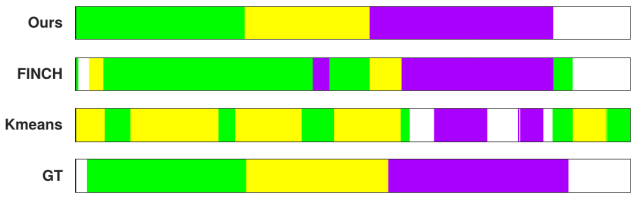


Figure 2. Segmentation output on a video from Breakfast Dataset [14]: Our method provides more accurate segment lengths of actions occurring in this video.

Algo. 1 produces a hierarchy of partitions where each successive partition has fewer clusters. To provide the required number of clusters K we choose a partition in this hierarchy with the minimal number of clusters that is $\geq K$. If the selected partition has more than K clusters, we refine it, one merge at a time as outlined in Algo. 2, until K clusters ($i.e.$ actions) remain.

Note that since in each successive merge time-stamps represent the average or central time of the cluster, this automatically ensures that merged clusters are highly temporally consistent. This aspect of our proposal is important as it may provides better temporal ordering of actions. In temporal video segmentation, obtaining correct ordering of actions is crucial and quite challenging. Existing SoTA unsupervised methods [18, 32, 28] employ expensive post-processing mechanisms such as Generalized Mallow models [24], Gaussian mixture models and Viterbi decoding to improve the ordering of their predicted action segments. In contrast, because of our temporal weighing, our clustering algorithm inherently produces time-consistent clusters, thus largely preserving the correct lengths of the actions occurring in a video. In Figure 2 we visualize the obtained action segments and their order (by mapping the obtained segments under Hungarian matching) on a sample video. This video depicts 4 ground-truth clusters and has $\approx$ 800 frames. The first step of Algo. 1 (lines 4-5) provides a partition of these 800 frames with 254 clusters. The successive merges of this partitioning produce 3 hierarchical partitions with 67, 20, 3 and 1 cluster(s) and the algorithm stops in only 4 steps. We then use Algo. 2 to obtain the required number of ground-truth action segments, 4, for this video. The partition with the minimal number of clusters ≥ 4 (in this case partition 3 with 20 clusters) is refined one merge at a time to produce the 4 clusters or action segments. Note that, in direct contrast to Kmeans and FINCH, our temporally-weighted clustering provides better action segments and also preserves their order in the video.

While we use a similar procedure for hierarchical merges as in FINCH [31], our work differs in scope and technical approach. In our proposal we build a temporally modulated 1-NN graph which, unlike FINCH, requires us to use all pairwise distances of samples both in space and in time for building the adjacency matrix. Our method, thus, can be

Algorithm 1 Temporally Weighted Clustering Hierarchy

Input: Video $X = \{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}\}$, $X \in \mathbb{R}^{N \times d}$
Output: Set of Partitions $\mathcal{S} = \{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_S\}$ such that $\mathcal{P}_{i+1} \supseteq \mathcal{P}_i \ \forall i \in \mathcal{S}$. Each partition $\mathcal{P}_i = \{C_1, C_2, \cdots, C_{\mathcal{P}_i}\}$ is a valid clustering of X.
Initialization:
Initialize time-stamps $T = \{1, 2, \cdots, N\}$. Compute 1-NN temporally weighted graph G via Equation 1-4
Get first partition $\mathcal{P}_1$ with $C_{\mathcal{P}_1}$ clusters from connected-components of G .
while there are at least two clusters in $\mathcal{P}_i$ **do**
    Given input data X and its partition $\mathcal{P}_i$ prepare averaged data matrix $M = \{\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \cdots, \bar{\mathbf{x}}_{C_{\Gamma_i}}\}$ and averaged time-stamps $T_M = \{\bar{t}_1, \bar{t}_2, \cdots, \bar{t}_{C_{\mathcal{P}_i}}\}$, where $\mathbb{M}^{C_{\mathcal{P}_i} \times d}$ and $\mathbb{T}_{\mathbb{M}}{}^{C_{\mathcal{P}_i} \times 1}$.
    Compute 1-NN temporally weighted graph G_M via Equation 1-4 with feature vectors in M and time-stamps in T_M.
    Get partition $\mathcal{P}_M$ of $\mathcal{P}_i$ from connected-components of G_M.
    if $\mathcal{P}_M$ has one cluster **then**
        break
    else
        Update cluster labels in $\mathcal{P}_i : \mathcal{P}_M \rightarrow \mathcal{P}_i$
    end if
end while

Algorithm 2 Final Action Segmentation

Input: # of actions K, Video $X = \{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}\}$ and a partition $\mathcal{P}_i$ from the output of Algorithm 1.
Output: Partition $\mathcal{P}_K$ with required number of action labels.
Merge two clusters at a time:
for steps = # of clusters in $\mathcal{P}_i$ - K **do**
    Initialize time-stamps $T = \{1, 2, \cdots, N\}$. Given input data X and its partition $\mathcal{P}_i$ prepare averaged data matrix $M = \{\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \cdots, \bar{\mathbf{x}}_{C_{\mathcal{P}_i}}\}$ and averaged time-stamps $T_M = \{\bar{t}_1, \bar{t}_2, \cdots, \bar{t}_{C_{\mathcal{P}_i}}\}$
    Compute 1-NN temporally weighted graph G_M via Equation 1-4
    $\forall G_M(i,j) = 1$ keep only one symmetric link (i,j) with the minimum temporal distance $W(i,j)$ obtained in Equation 3 and set all others to zero.
    Update cluster labels in $\mathcal{P}_i$: Merge corresponding i, j clusters in $\mathcal{P}_i$
end for

considered a special case of FINCH which is well suited for videos. Because of these differences, and for clarity in comparing both, we term our method **T**emporally **W**eighted **FI**rst **N**N **C**lustering **H**ierarchy (TW-FINCH). For action segmentation, TW-FINCH shows clear performances advantages over both Kmeans and FINCH, as we will show next in section 4.

4. Experiments

In this section, we first introduce the datasets, features, and metrics used to evaluate our TW-FINCH method, before comparing it both to baseline and SoTA approaches.

Datasets: We conduct experiments on five challenging and popular temporal action segmentation datasets, namely Breakfast (BF) [14], Inria Instructional Videos (YTI) [2], 50Salads (FS) [35], MPII Cooking 2 (MPII) [30], and Hollywood Extended (HE) [4]. As shown in Table 2, these 5 datasets cover a wide variety of activities (from cooking different types of meals to car maintenance), contain videos of varying lengths (from 520 frames on average to up to 11788 frames), and have different levels of average action granularity (from 3 up to 19).

Features: To ensure a fair comparison to related work,

	BF [14]	YTI [2]	FS [35]	MPII [30]	HE [4]
#Videos	1712	150	50	273	937
Avg. #Frames-per-video	2099	520	11788	10555	835
Feature Dim.	64	3000	64	64	64
#Activities (V)	10	5	1	67	16
Avg. #Actions-per-video (A)	6	9	19	17	3
Background	7%	63.5%	14.1%	29%	61%

Table 2. Statistics of datasets used in the experiments: Background refers to the % of background frames in a dataset.

we use the same input features that were used by recent methods [18, 32, 11, 22, 27]. Specifically, for the BF, FS, MPII, and HE datasets we use the improved dense trajectory (IDT) [38] features computed and provided by the authors of CTE [18] (for BF and FS) and SCT [11] (for MPII and HE). For YTI [2], we use the features provided by the authors, which are 3000-dimensional feature vectors formed by a concatenation of HOF [19] descriptors and features extracted from VGG16-conv5 [33]. For all datasets, we report performance for the full dataset, consistent with literature.

Metrics: To evaluate the temporal segmentation, we require a one-to-one mapping between the predicted segments and ground-truth labels. Following [18, 32, 22, 11, 27], we generate such a mapping using the Hungarian algorithm then evaluate with four metrics: (i) accuracy, calculated as the mean over frames (MoF); (ii) the F1-score; (iii) the Jaccard index, calculated as the intersection over union (IoU); and (iv) the midpoint hit criterion [29], where the midpoint of the predicted segment must be within the ground-truth. We report MoF and IoU for all datasets, and in addition F1-score for YTI and midpoint hit for MPII, as used in previous works. For all metrics, a higher result indicates better performance.

Evaluation Setup: Recent methods [1, 18, 32, 22, 36] all

Breakfast dataset				
Supervision	Method	IoU	MoF	T
Fully Sup.	HOGHOF+HTK [14]	—	28.8	✓
	TCFPN [9]	—	52.0	✓
	HTK+DTF w. PCA [15]	9.8	56.3	✓
	GMM+CNN [16]	36.1	50.7	✓
	RNN+HMM [17]	—	61.3	✓
	MS-TCN [10]	—	66.3	✓
	SSTDA [7]	—	70.2	✓
Weakly Sup.	ECTC [13]	—	27.7	✓
	GMM+CNN [16]	12.9	28.2	✓
	SCT [11]	—	30.4	✓
	RNN-FC [26]	—	33.3	✓
	RNN+HMM [17]	—	36.7	✓
	TCFPN [9]	24.2	38.4	✓
	NN-Vit. [28]	—	43.0	✓
	D3TW [6]	—	45.7	✓
	MuCon [34]	—	49.7	✓
	CDFL [22]	33.7	50.2	✓
Unsup. Baselines	Equal Split	21.9	34.8	✗
	Kmeans	23.5	42.7	✗
	FINCH	28.3	51.9	✗
Unsup.	Mallow* [32]	—	34.6	✓
	CTE* [18]	—	41.8	✓
	LSTM+AL [1]	—	42.9	✓
	VTE-UNET [36]	—	52.2	✓
	TW-FINCH	42.3	62.7	✗
Unsup.	**TW-FINCH (K=gt/video)**	44.1	63.8	✗

Table 3. Comparison on the Breakfast dataset [14] (* denotes results with Hungarian computed over all videos of an activity together). T denotes whether the method has a training stage on target activity/videos.

evaluate at ground-truth number of actions for an activity.

We adopt a similar approach and set K, for a video of a given activity, as the average number of actions for that activity. To provide an upper limit on the performance of our method we also evaluate with K set as the groundtruth of each video.

4.1. Comparison with baseline methods

As established in section 3, Kmeans [23] and FINCH [31] are strong baselines for temporal action segmentation. In this section we establish an additional baseline, which we call *Equal Split*, that involves simply splitting the frames in a video into K equal parts. It can be viewed as a temporal clustering baseline based only on the relative time-stamps of each frame. This seemingly trivial baseline is competitive for all datasets and actually outperforms many recent weakly-supervised and unsupervised methods for the BF (Table 3) and FS (Table 5) datasets. TW-FINCH, however, consistently outperforms all baselines by significant margins on all five datasets, as shown in Table 3 (BF), Table 4 (YTI), Table 5 (FS), Table 6 (MPII) and Table 7 (HE). We attribute these strong results to better temporal consistency and ordering of actions, which TW-FINCH is able to achieve due to temporal weighting.

4.2. Comparison with the State-of-the-art

We now compare TW-FINCH to current state-of-the-arts, discussing results for each of the 5 datasets in turn. However, as noted in [18] even though evaluation metrics are comparable to weakly and fully supervised approaches, one needs to consider that the results of the unsupervised learning are reported with respect to an optimal assignment of clusters to ground-truth classes and therefore report the best possible scenario for the task. For each dataset, we report IoU and MoF results for TW-FINCH. We report additional metrics when they are commonly used for a given dataset.

The column **T** in the tables denotes whether the method requires training on the target videos of an activity before being able to segment them. A dash indicates no known reported results.

Breakfast dataset (BF): BF contains an average of 6 actions per video, and 7% of frames in the dataset are background frames.

In Table 3 we report results on BF and compare TW-FINCH with recent state-of-the-art unsupervised, weakly-supervised and fully-supervised approaches. TW-FINCH outperforms all unsupervised methods, with absolute improvements of 10.5% over the best reported unsupervised method VTE-UNET and 19.8% over LSTM+AL [1]. Similarly TW-FINCH outperforms the best reported weakly-supervised method CDFL [22] with a $8.6/12.5\%$ gain on the IoU/MoF metrics.

Methods [18, 32] train a separate segmentation model for each activity, and set K to the maximum number of groundtruth actions for that activity. They then report results by computing Hungarian over all videos of one activity. Since we are clustering each video separately, using K as maximum would over segment most of the videos. This however still enable us to show the impact on performance in such a case. When we set K to the maximum # actions per activity on the BF dataset, our performance is 57.8%, as many of the videos are over segmented. To see the purity of these over-segmented clusters we computed the weighted cluster purity in this setting, which comes out to be 83.8%. This high purity indicates that, even with an inexact K, our clusters can still be used for downstream tasks such as training self-supervised video recognition models.

Inria Instructional Videos (YTI): YTI contains an average of 9 actions per video, and 63.5% of all frames in the dataset are background frames.

In Table 4 we summarize the performance of TW-FINCH on YTI and compare to recent state-of-the-art unsupervised and weakly-supervised approaches. To enable direct comparison, we follow previous works and remove a

Inria Instructional Videos				
Supervision	Method	F1-Score	MoF	T
Unsup. Baselines	Equal Split	27.8	30.2	✗
	Kmeans	29.4	38.5	✗
	FINCH	35.4	44.8	✗
Unsup.	Mallow* [32]	27.0	27.8	✓
	CTE* [18]	28.3	39.0	✓
	LSTM+AL [1]	39.7	—	✓
	TW-FINCH	48.2	56.7	✗
Unsup.	**TW-FINCH (K=gt/video)**	51.9	58.6	✗

Table 4. Comparison on the Inria Instructional Videos [2] dataset. * denotes results with Hungarian computed over all videos of an activity together.

50Salads				
Supervision	Method	eval	mid	T
Fully Sup.	ST-CNN [21]	68.0	58.1	✓
	ED-TCN [20]	72.0	64.7	✓
	TricorNet [8]	73.4	67.5	✓
	MS-TCN [10]	80.7	—	✓
	SSTDA [7]	83.8	—	✓
Weakly Sup.	ECTC [13]	—	11.9	✓
	HTK+DTF [15]	—	24.7	✓
	RNN-FC [26]	—	45.5	✓
	NN-Vit. [28]	—	49.4	✓
	CDFL [22]	—	54.7	✓
Unsup. Baselines	Equal Split	47.4	33.1	✗
	Kmeans	34.4	29.4	✗
	FINCH	39.6	33.7	✗
Unsup.	LSTM+AL [1]	60.6	—	✓
	TW-FINCH	71.1	66.5	✗
Unsup.	**TW-FINCH (K=gt/video)**	71.7	66.8	✗

Table 5. Comparison to SoTA approaches at *eval* and *mid* granularity levels on the 50Salads dataset [35]. We report **MoF**.

ratio ($\tau = 75\%$) of the background frames from the video sequence and report the performance. TW-FINCH outperforms other methods and achieves F1-Score of 48.2% and MoF of 56.7%, which constitute absolute improvements of 8.5% on F1-Score over the best published unsupervised method.

Impact of Background on YTI. As 63.5% of all frames in the YTI dataset are background, methods that train on this dataset tend to over-fit on the background. In contrast, a clustering based method is not strongly impacted by this: when we evaluate TW-FINCH while including all of the background frames our MoF accuracy drops from 56.7 $\rightarrow$ 43.4% as is expected due to having more frames and thus more errors. Given such a significant data bias on background frames this relatively small drop indicates that TW-FINCH works reasonably with widely-varying degrees of background content.

50Salads (FS): FS contains an average of 19 actions per video, and 14.1% of all frames in the dataset are background

MPI Cooking 2						
Supervision	Method	IoU	Midpoint-hit Precision	Midpoint-hit Recall	MoF	T
Fully Sup.	Pose + Holistic [29]	—	19.8	40.2	—	✓
	Fine-grained [25]	—	28.6	54.3	—	✓
	GMM+CNN [16]	45.5	—	—	72.0	✓
Weakly Sup.	GMM+CNN [16]	29.7	—	—	59.7	✓
Unsup. Baselines	Equal Split	6.9	25.6	44.6	14.6	✗
	Kmeans	14.5	21.9	34.8	30.4	✗
	FINCH	18.3	26.3	41.9	40.5	✗
Unsup.	**TW-FINCH**	23.1	34.1	54.9	42.0	✗
Unsup.	**TW-FINCH (K=gt/video)**	24.6	37.5	59.2	43.4	✗

Table 6. Comparison on the MPII Cooking 2 dataset [30].

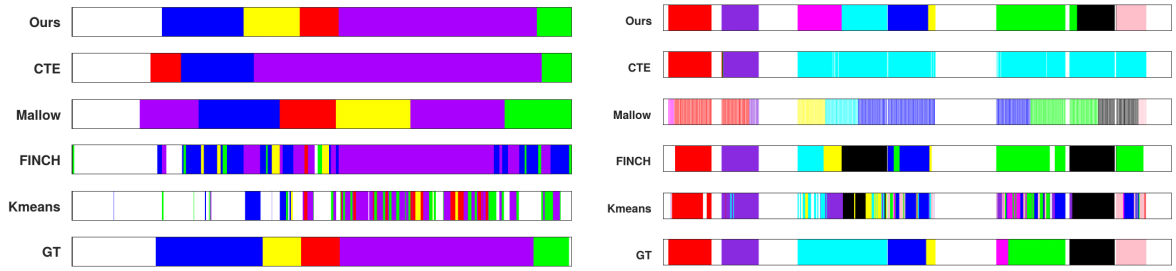
Hollywood Extended				
Supervision	Method	IoU	MoF	T
Fully Sup.	GMM+CNN [16]	8.4	39.5	✓
Weakly Sup.	GMM+CNN [16]	8.6	33.0	✓
	ActionSet [27]	9.3	—	✓
	RNN-FC [26]	11.9	—	✓
	TCFPN [9]	12.6	28.7	✓
	SCT [11]	17.7	—	✓
	D3TW [6]	—	33.6	✓
	CDFL [22]	19.5	45.0	✓
Unsup. Baselines	Equal Split	24.6	39.6	✗
	Kmeans	33.2	55.3	✗
	FINCH	37.1	56.8	✗
Unsup.	**TW-FINCH**	35.0	55.0	✗
Unsup.	**TW-FINCH (K=gt/video)**	38.5	57.8	✗

Table 7. Comparison on the Hollywood Extended dataset [4].

frames.We evaluate with respect to two action granularity levels, as described in [35]. The ***mid*** granularity level evaluates performance on the full set of 19 actions while the ***eval*** granularity level merges some of these action classes, resulting in 10 action classes. In Table 5 we show that TW-FINCH obtains a MoF of 66.5% in the *mid* granularity, 11.8% higher (in absolute terms) than the best weakly-supervised method CDFL [22]. We see similar performance gains in the *eval* granularity level evaluation as well. The IoU score of TW-FINCH for *mid* and *eval* granularity is 48.4% and 51.5% respectively.

MPII Cooking 2 (MPII): MPII contains 17 actions per video on average, and 29% of all frames in the dataset are background frames. For MPII we report the midpoint hit criterion [29] (multi-class precision and recall), the standard metric for this dataset, in addition to IoU and MoF. The dataset provides a fixed train/test split. We report performance on the test set to enable direct comparisons with previously reported results. As Table 6 shows, TW-FINCH outperforms our strong unsupervised baselines for all 4 reported metrics. Our method also outperforms SoTA fully-supervised methods that report the mid-point hit criterion.

Hollywood Extended (HE): HE contains an average of 3 (including background) actions per video, and 61% of all frames in the dataset are background frames. We report



(a) *P39_cam02_P39_scrambledegg* (b) *changing_tire_0023*

Figure 3. Segmentation examples from (a) the Breakfast dataset [14], and (b) the Inria Instructional Videos dataset [2]. Colors indicate different actions and are arranged in chronological order. We compare the segmentation quality of our method to Kmeans, FINCH, and two state-of-the-art unsupervised methods, CTE and Mallow. Our method has predicted better lengths of actions occurring in these videos.

results for HE in Table 7, which shows that TW-FINCH outperforms CDFL [22] by 15.5% (19.5→35.0) and 10.0% (45→55.0) in IoU and MoF, respectively. Further, note that the performance of our appearance-based clustering baselines is quite similar to the performance of our method. We attribute this to the small number of clusters per video (3 clusters on average). As a result, Kmeans and FINCH are roughly as effective as TW-FINCH, as temporally ordering 3 clusters is less difficult.

Qualitative Results. Fig. 3 shows representative results for two videos taken from the BF dataset (a) and the YTI dataset (b). Note that in the visualization (b) we set the background frames to white for all methods, according to the ground-truth. This allows the misclassification and mis-ordering of background frames to be more clearly seen. In (a), one can observe that TW-FINCH accurately predicts the length of segments, yielding better segmentation boundaries. Both clustering baselines, neither of which leverage temporal information, have noisy segments. Other SoTA unsupervised methods either have inaccurate temporal boundaries, incorrect ordering of actions, or are missing actions altogether. In addition, background frames are more often misclassified and mis-ordered for competing methods. Similar observations can be made in (b), where we show qualitative segmentation results on a more challenging YTI video with 9 actions of varying lengths, and several interspersed background scenes.

Limitations. Two main limitations for our work exist, which are inherent to our unsupervised clustering-based approach. The first, illustrated in Figure 3(b), occurs when we over-segment a temporally-contiguous sequence due to low visual coherence. The second may occur when we assign frames that depict the same action to different clusters because they are temporally distant.

Computational complexity. As we need to compute the $N \times N$ temporal distances, the computational complexity of TW-FINCH is $\mathcal{O}(N^2)$. In contrast FINCH is $\mathcal{O}(Nlog(N))$ while other similar graph-based clustering methods such as spectral methods are $\mathcal{O}(N^3)$ and hierarchical agglomerative linkage-based schemes are $\mathcal{O}(N^2log(N))$. Table 8 provides the total run-time of TW-FINCH and other state-of-the-art methods on Breakfast dataset split 1 (252 videos). Unlike previous methods that require hours of model training on GPUs, our method runs on a computer with an AMD 16-core processor, taking approximately 0.16 seconds on average to segment one video ($\approx$ 2000 frames).

Supervision	Method	Training (hours)	Testing (seconds)	T
Weakly Sup.	TCFPN* [9]	12.75	00.01	✓
	NN-Vit.* [28]	11.23	56.25	✓
	CDFL* [22]	66.73	62.37	✓
	MuCon-full* [34]	04.57	03.03	✓
Unsup. Baselines	Kmeans	00.00	38.69	✗
	FINCH	00.00	37.08	✗
Unsup.	CTE [18]	—	217.94	✓
	TW-FINCH (Ours)	00.00	40.31	✗

Table 8. Run-time comparison of method with other state-of-the-art methods on Breakfast dataset. Testing duration is measured as the average inference for split 1 test set (252 videos). *The run-time of all the Weakly Sup. methods were taken from [34]

5. Conclusion

We addressed the problem of temporal action segmentation and found that simple clustering baselines produce results that are competitive with, and often outperform, recent SoTA unsupervised methods. We then proposed a new unsupervised method, TW-FINCH which encodes spatiotemporal similarities between frames on a 1-nearest-neighbor graph and produces a hierarchical clustering of frames. Our proposal is practical as unlike existing approaches it does not require training on the target activity videos to produce its action segments. Our extensive quantitative experiments demonstrate that TW-FINCH is effective and consistently outperforms SoTA methods on 5 benchmark datasets by wide margins on multiple metrics.

References

[1] Sathyanarayanan N Aakur and Sudeep Sarkar. A perceptual prediction framework for self supervised event segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1197–1206, 2019.

[2] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4575–4583, 2016.

[3] Maryam Asadi-Aghbolaghi, Albert Clapés, Marco Bellantonio, Hugo Jair Escalante, Víctor Ponce-López, Xavier Baró, Isabelle Guyon, Shohreh Kasaei, and Sergio Escalera. Deep learning for action and gesture recognition in image sequences: A survey. In *Gesture Recognition*, pages 539–578. Springer, 2017.

[4] Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision*, pages 628–643. Springer, 2014.

[5] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[6] Chien-Yi Chang, De-An Huang, Yanan Sui, Li Fei-Fei, and Juan Carlos Niebles. D3tw: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3546–3555, 2019.

[7] Min-Hung Chen, Baopu Li, Yingze Bao, Ghassan AlRegib, and Zsolt Kira. Action segmentation with joint self-supervised temporal domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9454–9463, 2020.

[8] Li Ding and Chenliang Xu. Tricornet: A hybrid temporal convolutional and recurrent network for video action segmentation. *arXiv preprint arXiv:1705.07818*, 2017.

[9] Li Ding and Chenliang Xu. Weakly-supervised action segmentation with iterative soft boundary assignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6508–6516, 2018.

[10] Yazan Abu Farha and Jurgen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3575–3584, 2019.

[11] Mohsen Fayyaz and Jurgen Gall. Sct: Set constrained temporal transformer for set supervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 501–510, 2020.

[12] Samitha Herath, Mehrtash Harandi, and Fatih Porikli. Going deeper into action recognition: A survey. *Image and vision computing*, 60:4–21, 2017.

[13] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision*, pages 137–153. Springer, 2016.

[14] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 780–787, 2014.

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

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

[17] Hilde Kuehne, Alexander Richard, and Juergen Gall. A hybrid rnn-hmm approach for weakly supervised temporal action segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[18] Anna Kukleva, Hilde Kuehne, Fadime Sener, and Jurgen Gall. Unsupervised learning of action classes with continuous temporal embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12066–12074, 2019.

[19] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[20] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165, 2017.

[21] Colin Lea, Austin Reiter, René Vidal, and Gregory D Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. In *European Conference on Computer Vision*, pages 36–52. Springer, 2016.

[22] Jun Li, Peng Lei, and Sinisa Todorovic. Weakly supervised energy-based learning for action segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6243–6251, 2019.

[23] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[24] Colin L Mallows. Non-null ranking models. i. *Biometrika*, 44(1/2):114–130, 1957.

[25] Bingbing Ni, Vignesh R Paramathayalan, and Pierre Moulin. Multiple granularity analysis for fine-grained action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 756–763, 2014.

[26] Alexander Richard, Hilde Kuehne, and Juergen Gall. Weakly supervised action learning with rnn based fine-to-coarse modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 754–763, 2017.

[27] Alexander Richard, Hilde Kuehne, and Juergen Gall. Action sets: Weakly supervised action segmentation without ordering constraints. In *Proceedings of the IEEE Conference*

on Computer Vision and Pattern Recognition, pages 5987–5996, 2018.

[28] Alexander Richard, Hilde Kuehne, Ahsan Iqbal, and Juergen Gall. Neuralnetwork-viterbi: A framework for weakly supervised video learning. In *CVPR*, 2018.

[29] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. In *2012 IEEE conference on computer vision and pattern recognition*, pages 1194–1201. IEEE, 2012.

[30] Marcus Rohrbach, Anna Rohrbach, Michaela Regneri, Sikandar Amin, Mykhaylo Andriluka, Manfred Pinkal, and Bernt Schiele. Recognizing fine-grained and composite activities using hand-centric features and script data. *International Journal of Computer Vision*, 119(3):346–373, 2016.

[31] M. Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943, 2019.

[32] Fadime Sener and Angela Yao. Unsupervised learning and segmentation of complex activities from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8368–8376, 2018.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] Yaser Souri, Mohsen Fayyaz, Luca Minciullo, Gianpiero Francesca, and Juergen Gall. Fast weakly supervised action segmentation using mutual consistency. *arXiv preprint arXiv:1904.03116*, 2019.

[35] Sebastian Stein and Stephen J McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 729–738, 2013.

[36] Rosaura G VidalMata, Walter J Scheirer, and Hilde Kuehne. Joint visual-temporal embedding for unsupervised learning of actions in untrimmed sequences. *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[37] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 2007.

[38] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision*, pages 3551–3558, 2013.