Supplementary Materials: Weakly-Supervised Action Segmentation and Alignment via Transcript-Aware Union-of-Subspaces Learning

Zijia Lu
Northeastern Univeristy
lu.zij@northeastern.edu

Ehsan Elhamifar
Northeastern University
e.elhamifar@northeastern.edu

1. Comparisons of TASL with Prior Works

In this section, we report the results of NNV [9], D3TW [1], CDFL [7], TASL(3) and TASL(10,3) with the performance of the *best run* and *average over three runs* as *Best* and *Average*, respectively. Since D3TW has not release its code, we were unable to implement it, thus report its available *Best* results on Breakfast from [7]. Per our knowledge, D3TW does not release the results for the five metrics we use on Hollywood and CrossTask. Meanwhile, we also compare with the state-of-the-art unsupervised subspace clustering methods: KSUB [4], SSC [3], LRR [8] and DSC [5].

As the current unsupervised subspace clustering baselines cannot leverage transcripts, we provide a modification to adapt them to the weakly-supervised setting. For KSUB, SSC and LRR, **1)** we run the methods on each training video separately to obtain initial unsupervised segmentations (clustering) by setting the number of clusters equal to the length of a video's transcript $\mathcal{T} = \{a_1, \ldots, a_n\}$. Notice that the segmentations do not necessarily conform to transcripts, due to their unsupervised nature. **2)** For each video, we match the discovered clusters with the transcript. We consider the timestamp of a cluster as the average timestamps of its frames. Then the cluster with the i-th smallest timestamp is matched with a_i. For example, if cluster 1 contains frames $\{3, 5, 6\}$ and cluster 2 contains frames $\{1, 2, 4\}$, cluster 1 is matched with a_2 and cluster 2 is matched with a_1. **3)** With the matching of frames to actions obtained, we can learn an action subspaces for *the video* using the frames in the correspondent clusters. Then constrained Viterbi decoding can be performed to obtain a transcript-consistent segmentation by solving

$$\min_{\{l_i\}, \sum l_i = N} \sum_{i=1}^{n} \sum_{t=L_i+1}^{L_i+l_i} d(t, i), \tag{1}$$

where $d(t, i)$ is the distance between frame t and the subspace of a_i and $L_i \triangleq \sum_{j=1}^{i-1} l_j$ is the total length of actions before a_i. **4)** So far, we obtained transcript-consistent segmentations using per-video action subspaces. We aggregate the frames assigned to an action across all videos to learn a universal action subspace. With these universal subspaces, we perform the proposed alignment algorithm, by setting the objective function of (1) as the constrained Viterbi decoding's cost, to obtain segmentations of test videos for action segmentation and action alignment tasks.

Given that DSC is a deep subspace clustering method, we train the DSC network on all training videos and use it to obtain the initial unsupervised segmentations, which are subsequently used in steps 2–4 as explained above. Given that KSUB, DSC, SSC, LRR are either deterministic or obtain consistent results across different runs, so we did not report the average and best performance for them.

Notice that in Eq (4) in the main paper, the loss function terms related to the invalid alignments, $y^n_{t,a}(\log(\sigma_{t,a}) + \log(\psi_{t,a}))$, are unbounded from below and can reach negative infinity. We solve this issue in a similar fashion as in [7]: we discard easy negative alignments whose probabilities, $\{\sigma_{t,a}\}$, are close to zero. Therefore, the remaining terms are lower bounded.

Table 1 and 2 summarize the results. Notice that the performance of subspace clustering baselines are much lower than those of weakly-supervised methods. Although they has high MoF on Hollywood and CrossTask, this is simply because they oversegment majority of frames into background while more than 50% of the frames in these two datasets are background. As a result, their IoU-bg/IoD-bg are very low. The overall low performance of the subspace clustering baselines comes from the fact that they do not have a principled way of learning from transcripts. Moreover, they cannot model lengths and frequencies of actions, thus are unable to incorporate the constraint loss $\mathcal{L}_{reg}$. Notice that DSC, which has the ability to perform feature learning, does better than other subspace clustering baselines.

TASL achieves state-of-the-art performance on all datasets for both action segmentation and action alignment

tasks. On Breakfast, TASL(3) and TASL(10, 3) both have improved CDFL on *Best* and *Average* results. It shows TASL is effective in learning actions and more robust to different initializations. On Hollywood, TASL(10, 3) has the best overall performance, while TASL(3) has comparable results on IoD and IoD-bg. It is partially due to Hollywood bearing more differences in action complexity, as it includes actions like *fight person, drive car* as well as *sit down, sit up*. On CrossTask, TASL(3) is able to outperform CDFL by large margins. TASL(10, 3) has high Mof but lower other scores as it has a large subspace dimension and tends to overpredict the most frequent action, background. Comparing TASL(3) and TASL(10, 3) shows by a proper subspace dimension, our method can correctly capture the action complexity while avoiding overpredicting background.

2. Window Size for Generating Candidate Alignments

Our proposed alignment algorithm generates multiple valid alignments by shifting action boundaries. We set a window size, δ, to decide the range of shift similar to [7]. For example, for an initial action boundary at frame t, the shifted boundaries will be in $[t - \delta/2, t + \delta/2]$. We set $\delta = 10$ for Breakfast, 6 for Hollywood and 4 for CrossTask.

3. Centers vs Subspaces

Since our TASL framework allows choosing different subspace dimensions, one choice is to use a 0-dimensional subspace, hence, multiple centroids (the learned frame feature of each action must be close to a point). This would be equivalent to transcript-aware K-means, which thus can be considered as a special case of our framework. However, we found that zero-dimensional subspaces often give lower performance due to inadequate capacity and representation power to model each action. In Table 3 and 4 we report the performance of zero-dimensional subspaces on the Hollywood dataset. It can be observed that TASL(0) obtains a lower performance than TASL(10, 3), as the small subspace dimension limits its learning capacity. Thus, it is prone to overfitting and over-predicting the 'background' class and often has a high MoF but low IoU/IoD.

4. Effect of 'Background' Subspace Dimension

Table 5 shows the effect of different subspace dimension for the background class on Breakfast for the action segmentation task. The first two rows show the previous results of CDFL and TASL(3) on Breakfast. Then we change TASL(3) to model background with different subspace dimension, with $d_{bg} \in \{1, 5, 10\}$ and Q_{bg} being identity, and keep those of other actions intact. Notice that TASL(3), with $d_{bg} = 3$, achieves the best performance

Breakfast	Mof	IoU	IoU -bg	IoD	IoD -bg
KSUB[4]	8.0	0.8	0.5	3.0	2.7
SSC[3]	17.2	11.7	6.9	16.8	13.2
LRR[8]	16.5	11.1	6.2	16.2	12.3
DSC[5]	11.8	12.0	6.7	19.6	15.3
Best					
NNV [9]	42.9	32.2	29.1	32.1	31.8
D3TW [1]	45.7	-	-	-	-
CDFL [7]	**50.8**	35.7	33.6	46.8	45.7
TASL(3)	49.9	**36.6**	**34.3**	**47.7**	**46.4**
TASL(10,3)	49.7	36.5	34.0	47.6	45.9
Average					
NNV [9]	40.2	31.2	27.7	41.4	38.9
CDFL [7]	47.2	34.1	31.3	44.9	43.7
TASL(3)	**47.8**	**35.2**	32.6	**46.1**	**44.5**
TASL(10,3)	47.9	35.1	**32.7**	46.0	44.1

Hollywood	Mof	IoU	IoU -bg	IoD	IoD -bg
KSUB[4]	47.6	4.5	1.4	7.6	4.2
SSC[3]	50.2	19.2	0.7	23.3	1.3
LRR[8]	49.5	19.0	0.8	23.6	1.7
DSC[5]	47.4	18.9	0.9	23.7	1.8
Best					
NNV [9]	44.4	23.2	13.1	34.5	17.8
CDFL [7]	40.7	22.2	15.1	36.1	19.0
TASL(3)	45.6	23.6	13.8	35.4	18.7
TASL(10,3)	**46.6**	**25.2**	**15.3**	**37.7**	**21.3**
Average					
NNV [9]	43.1	22.2	11.8	33.7	16.2
CDFL [7]	39.9	21.6	**14.1**	35.3	18.0
TASL(3)	43.7	23.3	13.6	**35.7**	**18.3**
TASL(10,3)	**43.7**	**23.4**	13.6	**35.7**	**18.3**

CrossTask	Mof	IoU	IoU -bg	IoD	IoD -bg
KSUB[4]	49.3	10.7	0.0	14.7	0.0
SSC[3]	65.3	12.7	0.1	14.3	0.2
LRR[8]	66.6	13.1	0.1	14.4	0.2
DSC[5]	64.0	12.6	0.2	14.6	0.6
Best					
NNV [9]	27.0	11.0	8.5	24.4	10.1
CDFL [7]	32.5	11.8	7.7	24.0	9.6
TASL(3)	42.7	**14.9**	**9.2**	**25.5**	**11.3**
TASL(10,3)	**52.6**	12.3	3.6	19.6	4.7
Average					
NNV [9]	26.5	10.7	7.9	24.0	9.4
CDFL [7]	31.9	11.5	7.5	23.8	9.3
TASL(3)	40.7	**14.5**	**8.9**	**25.1**	**11.0**
TASL(10,3)	**49.2**	12.2	2.8	18.5	3.7

Table 1: Action Segmentation Accuracies on Three Datasets.

while a larger or smaller background dimension decreases the performance, showing $d_{bg} = 3$ is the most proper background dimension on Breakfast.

5. Complexity of the Alignment Algorithm

In Section 3.2 of the paper, we introduced our alignment algorithm, which involves two steps: 1) finding the opti-

Breakfast	Mof	IoU	IoU -bg	IoD	IoD -bg
KSUB[4]	13.8	12.5	10.9	26.3	27.2
SSC[3]	28.5	17.0	15.5	35.4	39.2
LRR[8]	24.7	15.1	13.6	30.3	33.2
DSC[5]	19.2	15.3	13.3	33.8	37.2
Best					
NNV [9]	59.5	47.0	47.7	61.7	65.0
D3TW [1]	57.0	-	-	56.3	-
CDFL [7]	**67.6**	50.5	51.3	65.1	**69.5**
TASL(3)	65.8	**51.0**	**51.9**	**65.5**	69.1
TASL(10,3)	65.6	50.6	51.4	65.4	68.8
Average					
NNV [9]	55.9	45.2	45.6	60.1	63.4
CDFL [7]	62.1	47.8	48.4	63.1	67.1
TASL(3)	**64.1**	**49.9**	**50.7**	**64.7**	**68.2**
TASL(10,3)	63.4	49.6	50.2	64.6	67.7

Hollywood	Mof	IoU	IoU -bg	IoD	IoD -bg
KSUB[4]	57.0	26.5	8.4	40.6	27.8
SSC[3]	60.0	27.5	10.1	41.9	29.9
LRR[8]	59.4	27.0	9.5	40.7	27.7
DSC[5]	57.3	27.1	9.3	41.6	29.3
Best					
NNV [9]	61.5	35.9	26.4	51.3	41.5
CDFL [7]	60.2	36.9	**31.5**	51.1	40.9
TASL(3)	63.0	37.6	29.4	52.3	42.0
TASL(10,3)	**63.7**	**38.3**	30.7	**53.2**	**43.0**
Average					
NNV [9]	59.8	35.0	25.4	49.9	39.6
CDFL [7]	59.5	36.5	**30.7**	51.7	40.2
TASL(3)	61.6	36.8	28.3	51.5	41.1
TASL(10,3)	**62.2**	**37.7**	30.0	**52.4**	**41.7**

CrossTask	Mof	IoU	IoU -bg	IoD	IoD -bg
KSUB[4]	64.7	14.9	3.1	20.8	8.7
SSC[3]	63.9	14.6	3.0	20.4	8.2
LRR[8]	64.1	14.7	3.2	20.5	8.4
DSC[5]	63.8	14.8	3.3	20.7	8.6
Best					
NNV [9]	34.6	15.3	11.4	27.5	14.0
CDFL [7]	46.7	17.2	11.5	28.0	14.5
TASL(3)	57.1	**19.1**	**11.7**	**28.9**	**15.8**
TASL(10,3)	**60.0**	18.4	8.7	27.0	14.1
Average					
NNV [9]	34.3	15.1	11.3	27.1	13.4
CDFL [7]	43.4	17.0	11.3	27.6	14.3
TASL(3)	54.6	**18.8**	**11.5**	**28.2**	**15.2**
TASL(10,3)	**58.9**	17.9	8.0	26.1	13.5

Table 2: Action Alignment Accuracies on Three Datasets.

mal alignment using constrained Viterbi decoding; 2) generating positive and negative soft alignments $\boldsymbol{Y}^p, \boldsymbol{Y}^n$ from multiple valid and invalid alignments. Since we employ the constrained Viterbi decoding, similar to [9, 7], the complexity for step 1 is $O(N^2 n)$ for a video with N frames and with n actions in the transcript.

Hollywood	Mof	IoU	IoU-bg	IoD	IoD-bg
Best					
TASL(0)	**46.7**	24.1	13.1	35.5	17.9
TASL(10, 3)	46.6	**25.2**	**15.3**	**37.7**	**21.3**
Average					
TASL(0)	**45.8**	23.0	12.0	34.8	17.0
TASL(10, 3)	43.7	**23.4**	**13.6**	**35.7**	**18.3**

Table 3: The performance of 0-dimensional subspace for action segmentation on Hollywood.

Hollywood	Mof	IoU	IoU-bg	IoD	IoD-bg
Best					
TASL(0)	63.5	37.9	28.7	**53.2**	42.5
TASL(10, 3)	**63.7**	**38.3**	**30.7**	**53.2**	**43.0**
Average					
TASL(0)	**62.8**	37.2	28.2	52.0	41.0
TASL(10, 3)	62.2	**37.7**	**30.0**	**52.4**	**41.7**

Table 4: The performance of 0-dimensional subspace for action alignment on Hollywood.

Breakfast	Mof	IoU	IoU-bg	IoD	IoD-bg
CDFL [7]	47.2	34.1	31.3	44.9	43.7
TASL(3)	**47.8**	**35.2**	**32.6**	**46.1**	**44.5**
$d_{bg} = 1$	45.5	32.9	29.9	43.8	42.4
$d_{bg} = 5$	44.5	32.7	30.0	43.6	42.6
$d_{bg} = 10$	44.3	31.7	28.6	42.5	41.1

Table 5: Effect of 'Background' subspace dimension for TASL(3) on Breakfast for action segmentation.

Next, we discuss the complexity of step 2 of our alignment method, which would be similar to the complexity of CDFL [7]. Recall that, as defined in Eq.9 of the main paper, $\boldsymbol{Y}^p$ is the weighted average of multiple valid alignments:

$$\boldsymbol{Y}^p \triangleq \sum_k \alpha_k \boldsymbol{R}_k^p, \quad \alpha_k \triangleq \frac{\exp(s(\boldsymbol{R}_k^p))}{\sum_j \exp(s(\boldsymbol{R}_j^p))}, \tag{2}$$

where $s(\boldsymbol{R}_k^p) \triangleq \langle \boldsymbol{R}_k^p, \boldsymbol{\Delta} \rangle$ is the log likelihood of the k-th valid alignment. $\boldsymbol{Y}^n$ is defined similarly based on the invalid alignments. As we generate exponentially many valid and invalid alignments following [7], iteratively calculating α_k for each alignment is prohibitive. We develop an efficient approach to directly compute $\boldsymbol{Y}^n$ and $\boldsymbol{Y}^n$. First, we compute a smooth maximum of the log likelihood of all valid alignments via the log-sum-exp trick:

$$r^p \triangleq \log \left[\sum_k \exp(s(\boldsymbol{R}_k^p)) \right], \tag{3}$$

where r^p can be computed in $O(A^2 \delta n)$ using the method in

[7]. A is the total number of actions and δ is the window size mentioned in Section 2. Then $\boldsymbol{Y}^p$ can be obtained by computing the derivative of r^p w.r.t $\boldsymbol{\Delta}$. Specifically,

$$\frac{\partial r^p}{\partial \boldsymbol{\Delta}} = \sum_k \frac{\partial r^p}{\partial s(\boldsymbol{R}_k^p)} \frac{\partial s(\boldsymbol{R}_k^p)}{\partial \boldsymbol{\Delta}} \tag{4}$$

$$= \sum_k \frac{\exp(s(\boldsymbol{R}_k^p))}{\sum_j \exp(s(\boldsymbol{R}_j^p))} \boldsymbol{R}_k^p. \tag{5}$$

Similarly, $\boldsymbol{Y}^n$ can be obtained from the derivative of $r^n \triangleq \log[\sum_k \exp(s(\boldsymbol{R}_k^n))]$. The derivative can be efficiently computed using Pytorch autograd module. Yet, explicitly writing out $\boldsymbol{Y}^p$, $\boldsymbol{Y}^n$ gives us a more flexible control over the network optimization, as we can compute $\boldsymbol{Y}^p$, $\boldsymbol{Y}^n$ based on $\sigma_{t,a}$, as in Eq.5 of the paper, and use them to optimize both $\sigma_{t,a}$ and $\psi_{t,a}$. One can also upscale or downscale $\boldsymbol{Y}^p$, $\boldsymbol{Y}^p$ to give higher weights to the important actions and lower weights to the unimportant ones. On Breakfast, on average, it takes 0.4s to obtain the optimal alignment, 0.1s to compute r^p, 7.2s for r^n then 0.4s to obtain $\boldsymbol{Y}^p$, $\boldsymbol{Y}^n$ by computing derivatives. In terms of inference, it on average takes 0.93s for NNV, 1.02s for CDFL and 1.04s for TASL to generate the segmentation for a test video for action segmentation task.

6. Hierarchical Segmentation Framework

In this section, we provide more details of our hierarchical segmentation method for the action segmentation task, proposed in Section 3.3 of the paper, which consists of three steps: 1) Use the facility location subset selection algorithm [6] to choose C representative transcripts from all training transcripts and assign each training transcript to one of the representative. 2) Use the constrained Viterbi decoding to align a test video with the C representative transcripts and find the best matching representative. 3) Align the video with the transcripts assigned to the optimal representative transcript to recover for the final alignment.

The facility location subset selection aims to find a set of at most C representative training transcripts that have the smallest distances to all training transcripts, thus, can best represent all training transcripts, i.e., it solves

$$\max_{\Lambda \subset \mathcal{V}, |\Lambda| \leq C} f(\Lambda) \text{ where } f(\Lambda) \triangleq \sum_{i=1}^{V} \max_{j \in \Lambda} d(\mathcal{T}_i, \mathcal{T}_j). \tag{6}$$

Here, $\mathcal{V} = \{1, \ldots, V\}$ stores the indices of all V training transcripts and Λ will contain indices of representative transcripts while $f(\Lambda)$ measures the sum of distances of every training transcript to its closest representative transcript. $d(\mathcal{T}_i, \mathcal{T}_j) = 2 \times edit(\mathcal{T}_1, \mathcal{T}_2)/(|\mathcal{T}_1| + |\mathcal{T}_2|)$ is the pairwise distance between $\mathcal{T}_i, \mathcal{T}_j$ that equals to the Levenshtein distance between the two transcripts normalized by their lengths. Please note we remove 'background' class from transcripts when computing their pairwise distances, as two transcripts can share the exact same key actions but with backgrounds at very different locations.

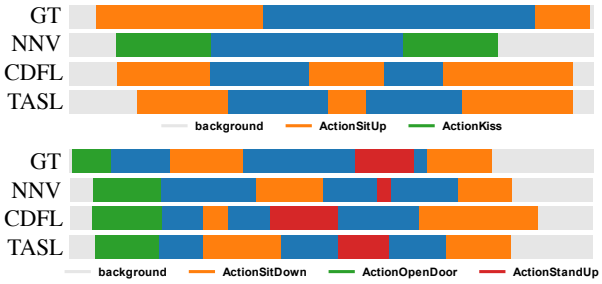

Figure 1: Results of NNV, CDFL, TASL(10, 3) against ground-truth, on two Hollywood videos for action segmentation (top) and alignment (bottom).

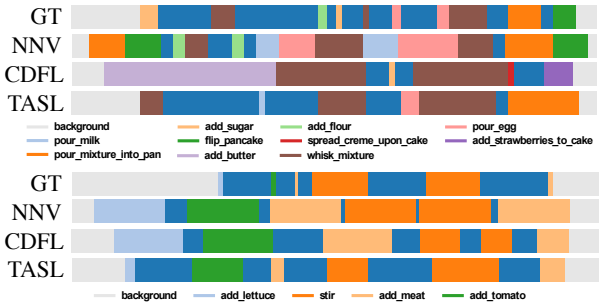

Figure 2: Results of NNV, CDFL, TASL(3) against ground-truth, on two CrossTask videos for action segmentation (top) and alignment (bottom).

Direct optimization of $f(\Lambda)$ over all subsets of size at most C is an NP-problem. An approximate method to solve this problem is a greedy algorithm that starts by initializing Λ as an empty set and, over C iterations, incrementally adds to Λ the index of a transcript that improves $f(\Lambda)$ the most. Algorithm 1 shows the steps of the greedy algorithm. After Λ is discovered, each training transcript will be assigned to its closest representative transcript, hence, we obtain clustering of transcripts as well.

With the representative transcripts and their groups of similar training transcripts, we first perform the constrained Viterbi decoding between a test video and the representative transcripts to find the one that gives the optimal alignment $\boldsymbol{R}_k^p$ with minimum $s(\boldsymbol{R}_k^p)$. This representative best matches with the video, thus its group of transcripts is most likely to contain the correct transcript of the test video. In the last step, we align the video with each training transcript in the representative transcript's group and return the optimal $\boldsymbol{R}_k^p$ with minimum $s(\boldsymbol{R}_k^p)$.

The performance of our method is robust to the number of representative transcripts, C. In Table 6, we tested

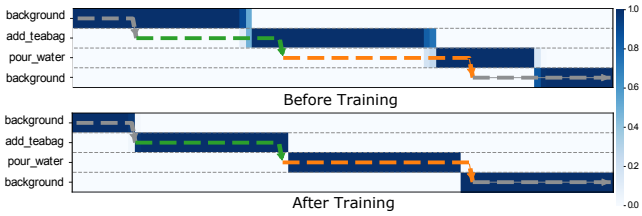

Figure 3: Positive alignment, $\boldsymbol{Y}^p$, generated by TASL(3) before and after training, against the ground-truth (colored lines), on a Breakfast video.

$C = \{5, 10, 20, 50, 100\}$ and reported the average MoF for action segmentation task on Breakfast. It can be observed that the change in performance is within 1% MoF, with $C = 20$ giving the best trade-off between performance and running time.

Algorithm 1: Greedy Algorithm

Initialize $\Lambda = \varnothing$;
for $j = 1, \ldots, C$ **do**
  for $i \in \mathcal{V} \backslash \Delta$ **do**
    Compute $\delta_f(i|\Lambda) \triangleq f(\Lambda \cup \{i\}) - f(\Lambda)$;
  end
  Compute $i^* = \text{argmax}_{i \in \mathcal{V} \backslash \Lambda} \delta_f(i|\Lambda)$;
  Update $\Lambda \leftarrow \Lambda \cup \{i^*\}$;
end

7. IoU and IoD Variations

As we mentioned in Section 4.1 of the paper, some works have used different ways of calculating IoU/IoD, leading to different performance ranges. In this section, we explain differences between ours (and existing works) and [2].

Given a test video with T frames, let $Y \in \{1, \ldots, |\mathcal{A}|\}^T$ denote the ground-truth (GT) framewise label of the video, where $\mathcal{A}$ is the set of unique actions in Y. Meanwhile, Y can also be encoded as a list of action segments, S^Y. Each segment is a tuple of three values: the action of the segment, its start timestamp and its end timestamp. Similarly, P and S^P will denote the predicted framewise label and action segments of the video.

Algorithm 2 and 3 show the pseudo-python code of [2]'s method and ours. Specifically, [2] compares the overlap between each GT and predicted segments. Yet, when a GT segment overlaps with multiple predicted segments, it only considers the predicted segment that gives the highest IoU/IoD and ignores others, thus often obtains an inflated score. Fig 4 shows a failure case for IoD using the metric of [2]: The overlap between the blue (orange) GT segment with respect to only the first (second) predicted blue segment, ignoring the second (first) one. Thus, one would obtain the falsely IoD of **1** despite an incorrect solution. To mitigate the issue, our method does not find matching between segments but directly compares the overlap between the GT and predicted frames of an action from all segments, i.e. the overlap between all GT and predicted blue/orange frames in Fig. 4. Consequently, our method obtains the IoD of **0.77**.

C	5	10	20	50	100
MoF	39.8	39.7	40.7	40.9	40.8

Table 6: Effect of the number of representative transcripts, C, on Breakfast for action segmentation.

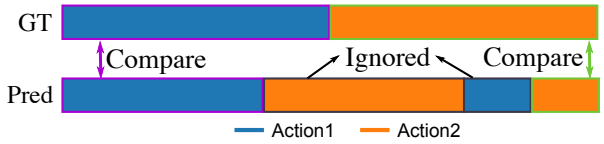

Figure 4: A failure case where using [2] leads to obtaining an IoD of 1. The first row is the ground-truth (GT) segmentation and the second row is predicted segmentation.

For fairness, we include the *Average* IoU/IoD scores from both ours and [2]'s method in Table 7. Due to the issue described above, [2]'s IoU/IoD are easier metrics and, in general, give inflated scores, especially for IoD. For example, it increases IoD by 6-10% on all datasets and doubles the IoD-bg on the CrossTask dataset.

Note on CDFL Results. In Table 1 and 2 of the main paper, the *Best* results on Hollywood of our implemented CDFL slightly differs from those reported in its paper. Since we used the released code from the authors, we hypothesize the difference could be due to train/test splits and to IoU/IoD calculation, which are not included in the code. In terms of train/test splits, We randomly generated four train/test splits, each with 90% videos as training data and 10% as test data, similar to the splits on Breakfast to make the experiments consistent across datasets. The exact ratio between the training/test videos used by the CDFL paper is unknown. Yet, it is important to note that we use exactly the same hyper-parameters for NVV/CDFL/TASL in all experiments and do not specifically tune the hyperparameters for TASL.

8. Qualitative Results

Figure 1 shows more results of NNV, CDFL and TASL against the ground-truth (GT) on Hollywood for action segmentation and action alignment tasks. Figure 2 shows similar results on CrossTask. On both datasets, TASL is able to better predict the action locations than NNV and CDFL. Specifically, in the action segmentation task, TASL discovers a transcript more similar to the correct one.

Algorithm 2: IoU/IoD from [2]

Data: S^Y, S^P: ground-truth and predicted segments of a test video

Initialize V as a list of zeros, with length equal to $|S^Y|$;

for $i,\ (a, t^{start}, t^{end})$ in enumerate(S^Y) **do**

  for $\hat{a}, \hat{t}^{start}, \hat{t}^{end}$ in S^P **do**

    if $a \neq \hat{a}$ **then** continue;

    if compute IoU **then**

      $v \leftarrow \frac{\min(t^{end},\hat{t}^{end}) - \max(t^{start},\hat{t}^{start})}{\max(t^{end},\hat{t}^{end}) - \min(t^{start},\hat{t}^{start})}$;

    else if compute IoD **then**

      $v \leftarrow \frac{\min(t^{end},\hat{t}^{end}) - \max(t^{start},\hat{t}^{start})}{\hat{t}^{end} - \hat{t}^{start}}$;

    end

  end

  $V[i] \leftarrow \max(V[i], v)$;

end

return mean(V)

Algorithm 3: IoU/IoD of our implementation

Data: $Y, P \in \mathcal{R}^T$: the ground-truth and predicted framewise label for a test video of length T.

Data: $\mathcal{A}$: list of actions in the video.

Initialize V as a list of zeros, with length equal to $|\mathcal{A}|$;

for i, a in enumerate($\mathcal{A}$) **do**

  if compute IoU **then**

    $V[i] \leftarrow \frac{\sum_t \mathbb{1}(Y_t=a \text{ and } P_t=a)}{\sum_t \mathbb{1}(Y_t=a \text{ or } P_t=a)}$;

  else if compute IoD **then**

    $V[i] \leftarrow \frac{\sum_t \mathbb{1}(Y_t=a \text{ and } P_t=a)}{\sum_t \mathbb{1}(P_t=a)}$;

  end

end

return mean(V)

Figure 3 shows the positive alignment $\boldsymbol{Y}^p$ generated by TASL(3) before and after training on a video in Breakfast. The colored line represents the ground-truth. Before training, the alignments are roughly uniform, thus, $\boldsymbol{Y}^p$ does not match the ground-truth. After multiple learning iterations, TASL produces more accurate $\boldsymbol{Y}^p$, which in turn improves its training. In the end, $\boldsymbol{Y}^p$ correctly aligns with the ground-truth. More importantly, in the plots of 'Before Training', light-blue blocks around the action boundaries show soft gradual transitions between adjacent actions, as $\boldsymbol{Y}^p$ combines alignments with slight shifts in action boundaries. This prevents TASL from overfitting to a poor initial alignment. While [2] uses also soft boundaries, theirs are handcrafted and fixed, while ours are controlled by $s(\boldsymbol{R})$ generated by the network. As our network becomes confident during training, the light-blue blocks disappear.

	IoU	IoU-bg	IoD	IoD-bg
Breakfast				
TASL(3)	35.2	32.6	46.1	44.5
TASL(3)†	36.1	32.1	55.5	40.9
Hollywood				
TASL(10,3)	23.4	13.6	35.7	18.3
TASL(10,3)†	26.3	12.5	43.2	23.0
CrossTask				
TASL(3)	14.5	8.9	25.1	11.0
TASL(3)†	21.3	9.1	35.1	26.1

(a) Action Segmentation

	IoU	IoU-bg	IoD	IoD-bg
Breakfast				
TASL(3)	49.9	50.7	64.7	68.2
TASL(3)†	49.4	50.6	70.2	64.3
Hollywood				
TASL(10,3)	37.7	30.0	52.4	41.7
TASL(10,3)†	38.0	29.9	60.1	47.7
CrossTask				
TASL(3)	18.8	11.5	28.2	15.2
TASL(3)†	23.2	12.3	37.3	31.1

(b) Action Alignment

Table 7: IoU/IoD from different calculation methods. For each dataset, the first row shows the scores from our method. The second row marked with † is from [2]'s method.

Figures 5, 6, 7 show, respectively, the largest, middle, smallest principal angles between subspaces learned by TASL(3) on all Breakfast actions. It can be observed that the largest angles between any two subspaces are almost all 90%, ensuring the subspaces are mutually orthogonal thus the learned features following the subspace structure will be discriminative. From the middle to smallest angles, we could gradually observe dark red blocks on the plots, each representing a group of semantically-relevant actions with small subspace angles between them. For example, the upper-right block, starting from *take plate* to *put pancake2plate*, are all actions from recipe *make pancake*. Surrounding this block are other *fry*-related actions but not necessarily from the *make pancake* recipe.

References

[1] 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. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2, 3

[2] Li Ding and Chenliang Xu. Weakly-supervised action seg-

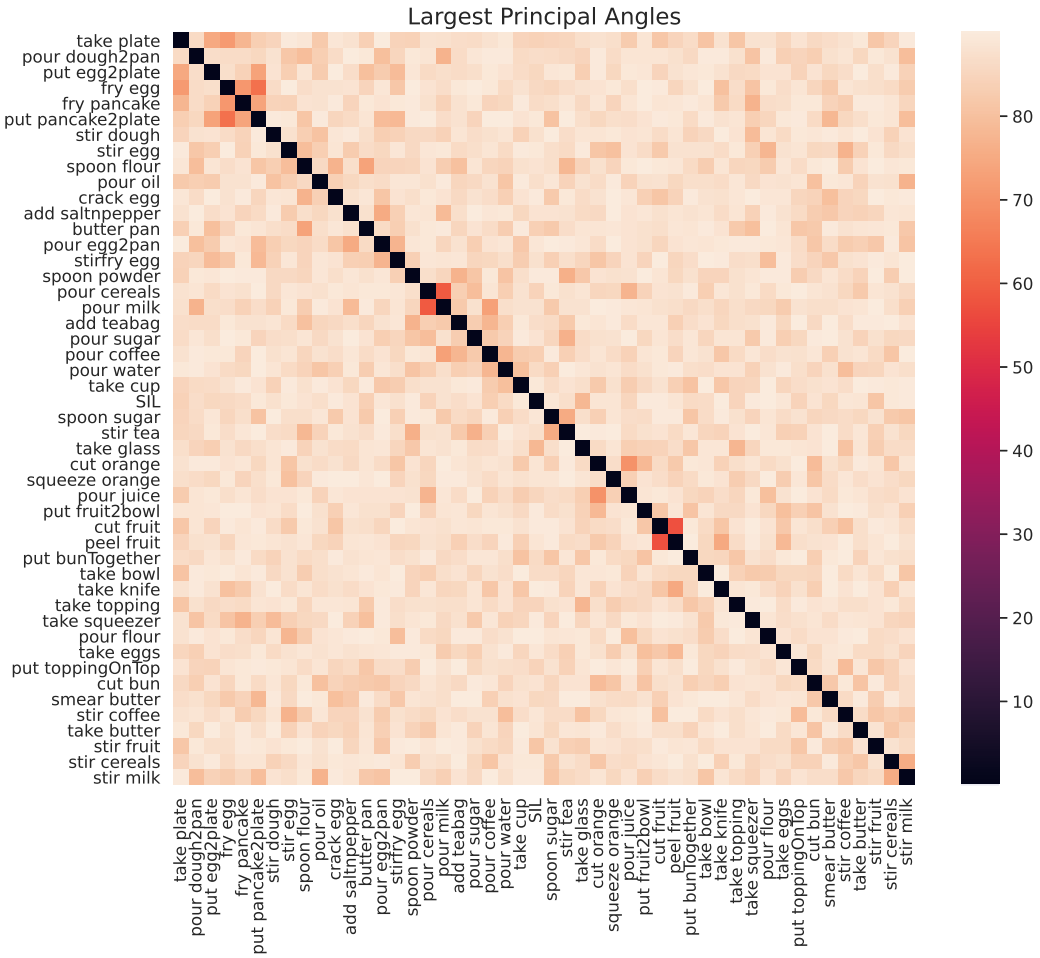

Figure 5: The largest principal angles between subspaces learned by TASL(3) on all Breakfast actions.

mentation with iterative soft boundary assignment. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 5, 6

[3] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013. 1, 2, 3

[4] J. Ho, M. H. Yang, J. Lim, K.C. Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2003. 1, 2, 3

[5] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid. Deep subspace clustering networks. *Neural Information Processing Systems*, 2017. 1, 2, 3

[6] Andreas Krause and Daniel Golovin. Submodular function maximization. Cambridge University Press, 2014. 4

[7] J. Li, P. Lei, and S. Todorovic. Weakly supervised energy-based learning for action segmentation. *International Conference on Computer Vision*, 2019. 1, 2, 3, 4

[8] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012. 1, 2, 3

[9] A. Richard, H. Kuehne, A. Iqbal, and J. Gall. Neuralnetwork-viterbi: A framework for weakly supervised video learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 2, 3

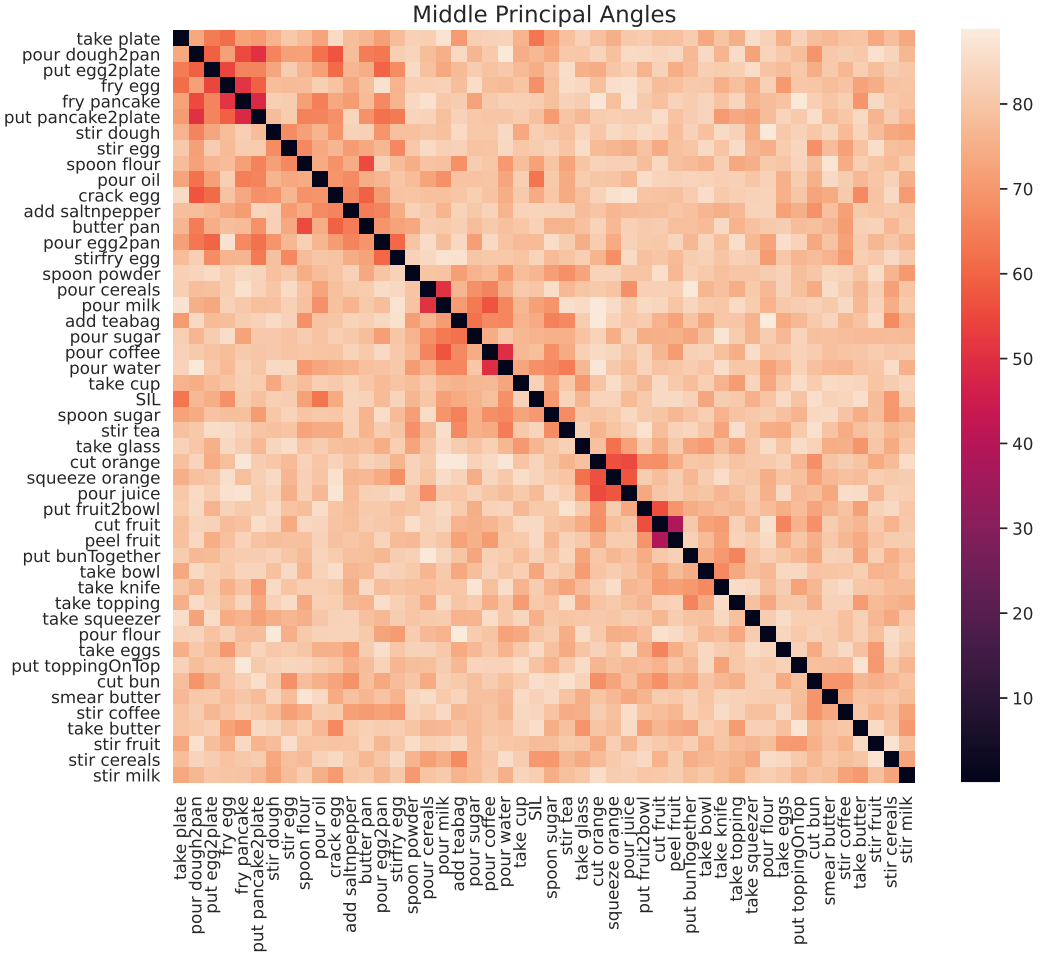

Figure 6: The middle principal angles between subspaces learned by TASL(3) on all Breakfast actions.

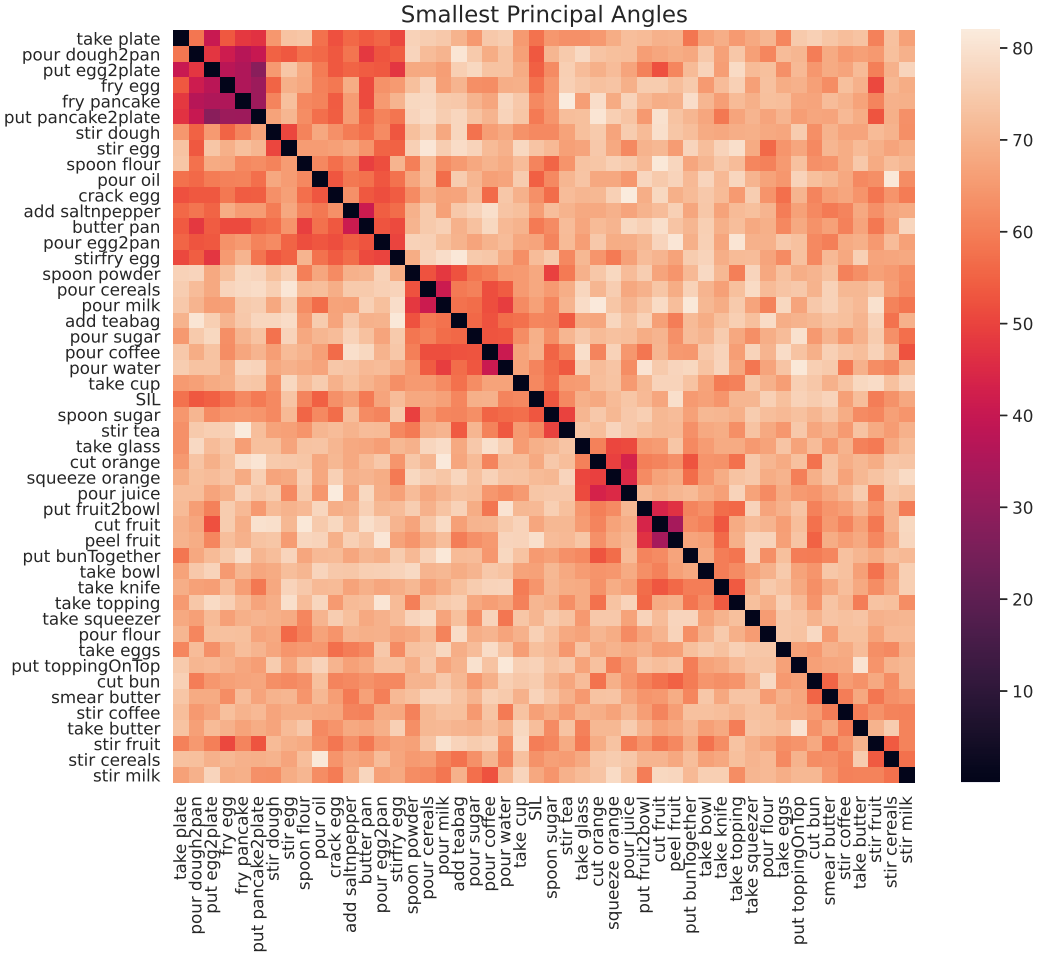

Figure 7: The smallest principal angles between subspaces learned by TASL(3) on all Breakfast actions.