Supplementary Material: Permutation-Aware Activity Segmentation via Unsupervised Frame-to-Segment Alignment

Quoc-Huy Tran* Ahmed Mehmood* Muhammad Ahmed Muhammad Naufil
Anas Zafar Andrey Konin M. Zeeshan Zia

Retrocausal, Inc., Redmond, WA
www.retrocausal.ai

This supplementary material begins with showing some qualitative results in Sec. S1. Next, we present the ablation results of using MLP encoder and using $\boldsymbol{A}$ in segment-/alignment-level modules in Secs. S2 and S3 respectively, and adopt the video-level segmentation method of ABD [2] for the activity-level segmentation task in Sec. S4. Finally, Sec. S5 provides the details of our implementation, while Sec. S6 includes a discussion on the societal impacts of our work.

S1. Qualitative Results

Fig. S1 illustrates the segmentation results of our approach and TOT [4] on two 50 Salads videos (*Eval* granularity). From Fig. S1, UFSA shows superior performance in extracting the permutation of actions. For example, let us consider the *'Add vinegar'* action (highlighted by red boxes) which happens at different temporal positions in the videos, UFSA captures the permutation of actions correctly, while TOT [4] maintains the fixed order of actions and hence fails to recognize the permutation of actions. Next, for actions that are missing, such as the *'Peel cucumber'* action, which occurs in Fig. S1a but does not appear in Fig. S1b, UFSA associates a negligible number of frames with this action class, whereas TOT [4] incorrectly assigns a large number of frames (highlighted by a green box).

Moreover, we include in Fig. S2 the segmentation results of our approach, TOT [4], and CTE [3] on other datasets, namely YouTube Instructions, Breakfast, and Desktop Assembly (*Orig* set). It is evident from Fig. S2 that our segmentation results are consistently closer to the ground truth than those of TOT [4] and CTE [3].

Nevertheless, our approach has a limitation in handling repetitive actions. For example, let us look at the *'Cut'* action in Fig. S1, which includes *'Cut tomato'*, *'Cut cucumber'*, *'Cut cheese'*, and *'Cut lettuce'* and hence occurs multiple times in the videos, our approach merges the multiple occurrences into a large segment (highlighted by blue boxes) since it assumes each action can happen only once. In addition, although TOT [4] has the same drawback, our combined segments are closer to the ground truth.

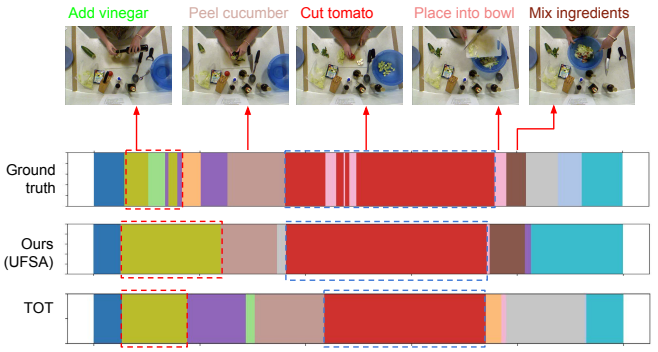
* indicates joint first author.
{huy,ahmedraza,ahmed,naufil,anas,andrey,zeeshan}@retrocausal.ai.

	Encoder	Decoder	MOF	F1
Eval	MLP	-	47.4	31.8
	Transformer	-	43.1	34.4
	MLP	Transformer	*47.8*	*34.8*
	Transformer	Transformer	**55.8**	**50.3**
YTI	MLP	-	40.6	30.0
	Transformer	-	42.8	30.2
	MLP	Transformer	*43.2*	*30.5*
	Transformer	Transformer	**49.6**	**32.4**

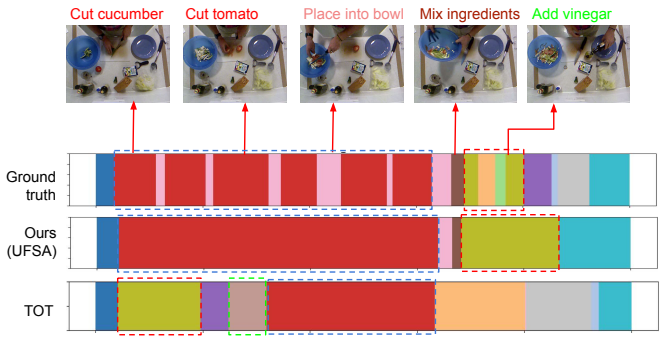
Table S1. Ablation with MLP encoder. Best results are in **bold**, while second best ones are *underlined*.

S2. Ablation with MLP encoder

We now perform an ablation study by using MLP encoder (instead of transformer encoder). Tab. S1 presents results on 50 Salads (*Eval* granularity) and YTI datasets. From the results, transformer encoder alone performs similarly as MLP encoder alone (i.e., TOT). Next, MLP encoder+transformer decoder yields small improvements over TOT as features extracted by MLP encoder do not capture contextual cues that are useful for transformer decoder. Lastly, large improvements over TOT are achieved when transformer encoder is used jointly with transformer decoder (i.e., our complete model).



(a) 50 Salads (*rgb-21-01*).



(b) 50 Salads (*rgb-15-02*).

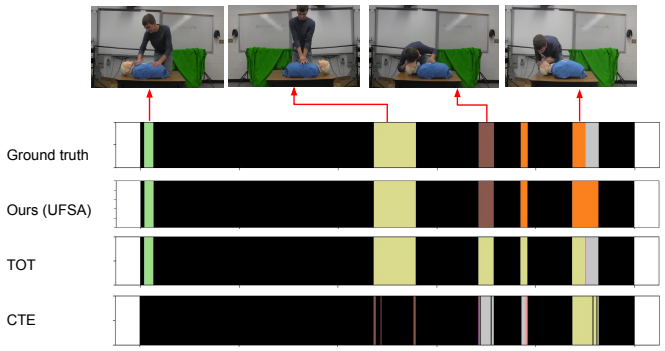
Figure S1. Segmentation results on two 50 Salads videos (*Eval* granularity). Red boxes highlight permuted actions. Green boxes highlight missing actions. Blue boxes highlight repetitive actions.

	Method	MOF	F1
Eval	Frame	43.1	34.4
	Frame+Segment	*43.3*	*37.8*
	Frame+Segment+Alignment	**46.1**	**45.2**
YTI	Frame	42.8	*30.2*
	Frame+Segment	*43.3*	**30.5**
	Frame+Segment+Alignment	**44.3**	29.4

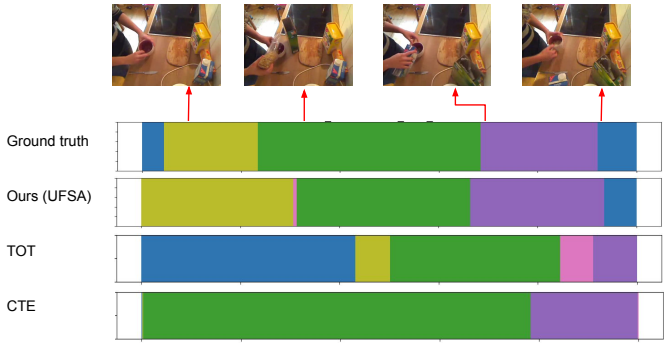
Table S2. Using $\boldsymbol{A}$ in segment-/alignment-level modules. Best results are in **bold**, while second best ones are *underlined*.

S3. Using $\boldsymbol{A}$ in segment-/alignment-level modules

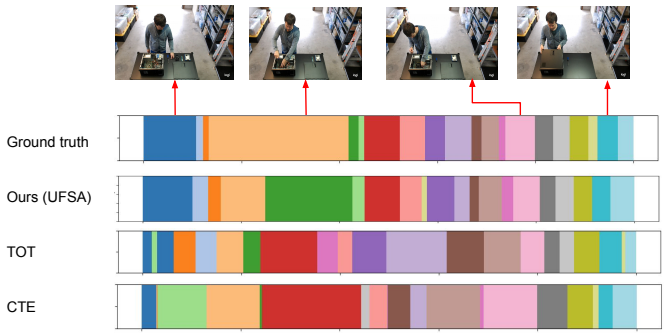
In this section, we repeat the ablation experiment in Tab. 1 of the main paper but we use the fixed-order prior $\boldsymbol{A}$ (instead of the permutation-aware prior $\boldsymbol{T}$) in segment-/alignment-level modules. Tab. S2 shows results on 50 Salads (*Eval* granularity) and YTI datasets. It can be seen from the results that using $\boldsymbol{A}$ in segment-/alignment-level modules improves results of frame-level module only, however, the improvements are smaller than those of using $\boldsymbol{T}$ (see Tab. 1 of the main paper).



(a) YouTube Instructions (*cpr_0027*).



(b) Breakfast (*P13_webcam01_P13_cereals*).



(c) Desktop Assembly (*2020-04-19_17-24-35*).

Figure S2. Segmentation results on (a) a YouTube Instructions video, (b) a Breakfast video, and (c) a Desktop Assembly video (*'Orig'* set).

S4. Comparisons with ABD [2]

Our method addresses the problem of activity-level segmentation, which jointly segments and clusters frames across all input videos. A related problem is video-level segmentation, which aims to segment a single input video only. Video-level segmentation is a sub-problem of activity-level segmentation and in general easier than activity-level segmentation. In this section, we evaluate the performance of a recent video-level segmentation method, i.e., ABD [2],

	Method	MOF	F1
Eval	⋆ABD [2]	**71.4**	-
	†ABD [2]	34.2	*32.8*
	†Ours (UFSA)	*55.8*	**50.3**
YTI	⋆ABD [2]	**67.2**	**49.2**
	†ABD [2]	29.4	29.4
	†Ours (UFSA)	*49.6*	*32.4*
Orig Breakfast	⋆ABD [2]	**64.0**	**52.3**
	†ABD [2]	23.6	21.7
	†Ours (UFSA)	*52.1*	*38.0*
Orig	⋆ABD [2]	*63.3*	*60.9*
	†ABD [2]	15.5	11.0
	†Ours (UFSA)	**71.2**	**72.2**
Extra	⋆ABD [2]	**60.8**	**57.1**
	†ABD [2]	12.0	10.6
	†Ours (UFSA)	*58.6*	*55.9*

Table S3. Comparisons with ABD [2]. Note that ⋆ denotes video-level results, whereas † denotes activity-level results. Best results are in **bold**, while second best ones are *underlined*.

for the task of activity-level segmentation. Firstly, for each input video, we run ABD [2] to obtain its video-level segmentation result. We then represent each segment in the result by its prototype vector, which is the average of feature vectors of frames belonging to that segment. Next, we perform K-Means clustering (K is set as the ground truth number of actions available in the activity) on the entire set of prototype vectors from all input videos to obtain the activity-level segmentation result, which we evaluate in Tab. S3. From the results, it can be seen that †UFSA outperforms †ABD [2] in the activity-level setting on all metrics and datasets. A more advanced clustering method which incorporates temporal information can be used instead of K-Means, however, it is out of the scope of our work. In addition, the video-level results of ⋆ABD [2] are mostly better than the activity-level results of †UFSA (except for Desktop Assembly - *Orig*), which is due to fine-grained video-level Hungarian matching [5].

S5. Implementation Details

Hyperparameter Settings. Tab. S4 presents a summary of our hyperparameter settings. For the temporal optimal transport problem in our frame-level prediction module and frame-to-segment alignment module, we follow the same hyperparameter settings used in TOT [4], including ρ and number of Sinkhorn-Knopp iterations. We keep the feature dimension d the same as TOT [4]. We use a single video, including all frames, per batch. In addition, for our transformer encoder and transformer decoder, we follow the same hyperparameter settings used in UVAST [1], including encoder dropout ratio and decoder dropout ratio. We set the temperature $\tau = 0.1$ (same as TOT [4]) in Sec. 3.1 of the main paper and the temperature $\tau' = 10^{-3}$ (same as UVAST [1]) in Sec. 3.3 of the main paper.

Computing Resources. All of our experiments are conducted with a single Nvidia A100 SXM4 GPU on Lambda Cloud.

S6. Societal Impacts

Our approach facilitates video recognition model learning without action labels, with potential applications in frontline worker training and assistance. Models generated from expert demonstration videos in various domains could offer guidance to new workers, improving the standard of care in fields such as medical surgery. However, at the same time, video understanding algorithms in surveillance applications may compromise privacy, even if it enhances security and productivity. Thus, we urge caution in the implementation of such technologies and advocate for the development of appropriate ethical guidelines.

References

[1] Nadine Behrmann, S Alireza Golestaneh, Zico Kolter, Jürgen Gall, and Mehdi Noroozi. Unified fully and timestamp supervised temporal action segmentation via sequence to sequence translation. In *European Conference on Computer Vision*, pages 52–68. Springer, 2022. 3

[2] Zexing Du, Xue Wang, Guoqing Zhou, and Qing Wang. Fast and unsupervised action boundary detection for action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3323–3332, 2022. 1, 2, 3

[3] Anna Kukleva, Hilde Kuehne, Fadime Sener, and Jurgen Gall. Unsupervised learning of action classes with continuous temporal embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12066–12074, 2019. 1

[4] Sateesh Kumar, Sanjay Haresh, Awais Ahmed, Andrey Konin, M Zeeshan Zia, and Quoc-Huy Tran. Unsupervised action segmentation by joint representation learning and online clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20174–20185, 2022. 1, 3

[5] Rosaura G VidalMata, Walter J Scheirer, Anna Kukleva, David Cox, and Hilde Kuehne. Joint visual-temporal embedding for unsupervised learning of actions in untrimmed sequences. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1238–1247, 2021. 3

Hyperparameter	**Value**
Temperature (τ)	0.1
Rho (ρ)	0.07 (**E**), 0.08 (**M**), 0.08 (**Y**), 0.05 (**B**), 0.07 (**O**), 0.07 (**A**)
Number of Sinkhorn-Knopp iterations	3
Feature dimension (d)	30 (**E**), 30 (**M**), 200 (**Y**), 40 (**B**), 30 (**O**), 30 (**A**)
Batch size	1
Learning rate	10^{-3}
Weight decay	10^{-5}
Number of encoder layers	2
Number of decoder layers	2
Encoder dropout ratio	0.3
Decoder dropout ratio	0.1
Temperature (τ')	10^{-3}

Table S4. Hyperparameter settings. **E** denotes 50 Salads (*Eval* granularity), **M** denotes 50 Salads (*Mid* granularity), **Y** denotes YouTube Instructions, **B** denotes Breakfast, **O** denotes Desktop Assembly (*Orig* set), and **A** denotes Desktop Assembly (*Extra* set).