Supplementary Material: Unsupervised Action Segmentation by Joint Representation Learning and Online Clustering

Sateesh Kumar† Sanjay Haresh† Awais Ahmed Andrey Konin
M. Zeeshan Zia Quoc-Huy Tran

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

In this supplementary material, we first discuss the limitations of our method in Sec. 1 and show some qualitative results in Sec. 2. Next, we provide the details of our implementation and our Desktop Assembly dataset in Secs. 3 and 4 respectively. Lastly, we discuss the societal impacts of our work in Sec. 5.

1. Limitation Discussions

Below we discuss the limitations of our method, including the equal partition constraint in Eq. 6 of the main text, the fixed order prior in Eq. 8 of the main paper, the performance of TCL, the comparison with ASAL, and the case of unknown activity class.

Equal Partition Constraint. We impose the equal partition constraint on cluster assignments, which may not hold true for the data in practice, i.e., one action might be longer than others in a given video. However, the equal partition constraint is imposed on soft cluster assignments (cluster assignment probabilities), i.e., the sum of soft cluster assignments should be equal for all clusters. More importantly, we apply the constraint at the mini-batch level (not the dataset level), which provides some flexibility to our approach, i.e., the sum of soft cluster assignments may be equal at the mini-batch level but the final cluster assignments may favor one cluster over others to some extent. For example, it may appear in Fig. 3(c) of the main paper that the soft cluster assignments are evenly distributed, but if we obtain the hard cluster assignments (by taking max over all soft cluster assignments for each frame), we observe that cluster #11 gets a slightly higher number of frames assigned than others. The above observations show that our approach may be capable of handling actions with various lengths to some extent, which is likely the case for the datasets used in this paper.

Fixed Order Prior. We apply a fixed order prior on the clusters learned via our approach. The fixed order prior allows us to introduce the temporal order-preserving constraint within the standard optimal transport module, and predict temporally ordered clusters which are more natural for video data and can be fed directly to the Viterbi decoding module at test time. As evident in Fig. 3 of the main text, OT without the fixed order prior fails to extract any temporal structure of the activity (see Fig. 3(a)), while TOT with the fixed order prior is able to capture the temporal order of the activity relatively well (see Fig. 3(c)), i.e., initial frames are assigned to cluster #1, following frames are assigned to cluster #2, subsequent frames are assigned to cluster #3, and so on. The ablation study results in Tabs. 1 and 2 of the main paper show that the fixed order prior provides performance gains on 50 Salads and YouTube Instructions, which further confirms the benefits of the fixed order prior. For the datasets used in this paper, permutation generally occurs when an action is not performed by the actor. In such cases, our method assigns only a few frames to the missing action (e.g., see the yellow segment in the TOT result in Fig. 4 of the main text) and hence manages to perform relatively well on the datasets used in this work. Nevertheless, we note that if there are several permutations or missing actions, our approach may not work.

TCL Performance. TCL has been used in many previous works, e.g., [2, 3, 8], to exploit temporal cues in videos for representation learning. Specifically, it encourages neighboring video frames to be mapped to nearby points in the embedding space (or belong to the same class) and distant video frames to be mapped to far away points in the embedding space (or belong to different classes). From our experiments above, TCL works well in cases of small/medium

† indicates joint first author.
{sateesh,sanjay,awais,andrey,zeeshan,huy}@retrocausal.ai.

intra-class variations, e.g., 50 Salads (*Mid* granularity), YTI, and Desktop Assembly datasets, while often not performing well in cases of large intra-class variations, e.g., 50 Salads (*Eval* granularity) and Breakfast datasets. Furthermore, our basic method (i.e., TOT) is able to achieve similar or better results than many previous methods on all datasets.

ASAL Comparison. On the Breakfast dataset, ASAL [7] performs the best, while our method (i.e., TOT) outperforms Mallow [9] and CTE [6] and performs on par with VTE [11] and UDE [10]. We note that ASAL is first initialized by CTE and then exploits action-level cues for refining the results of CTE (see Fig. 1 of the ASAL paper). Thus, we could instead utilize our method to provide a better initialization for ASAL and then leverage action-level cues with ASAL for boosting our performance. This remains an interesting direction for our future work. Furthermore, our method relies on a single two-layer MLP network (same as CTE), whereas ASAL employs a combination of three networks, i.e., two MLP networks and one RNN network. Since the objective of our work is to demonstrate the merit of an online clustering approach, we decide to use a single simple MLP network to facilitate a fair comparison with CTE (an offline clustering method).

Unknown Activity Class. Prior works and ours assume known activity classes and known number of actions per activity. To mitigate that, in Sec. 4.7 of CTE, it proposes to make *guesses* on values of K' (number of activity classes) and K (*same* number of actions per activity), and perform multi-level clustering to predict activity classes. However, the guesses are in fact very close to the ground truth ($K' * K = 50$ vs. ground truth 48). Our approach could be extended to perform multi-level clustering, but it is not trivial and remains our future work.

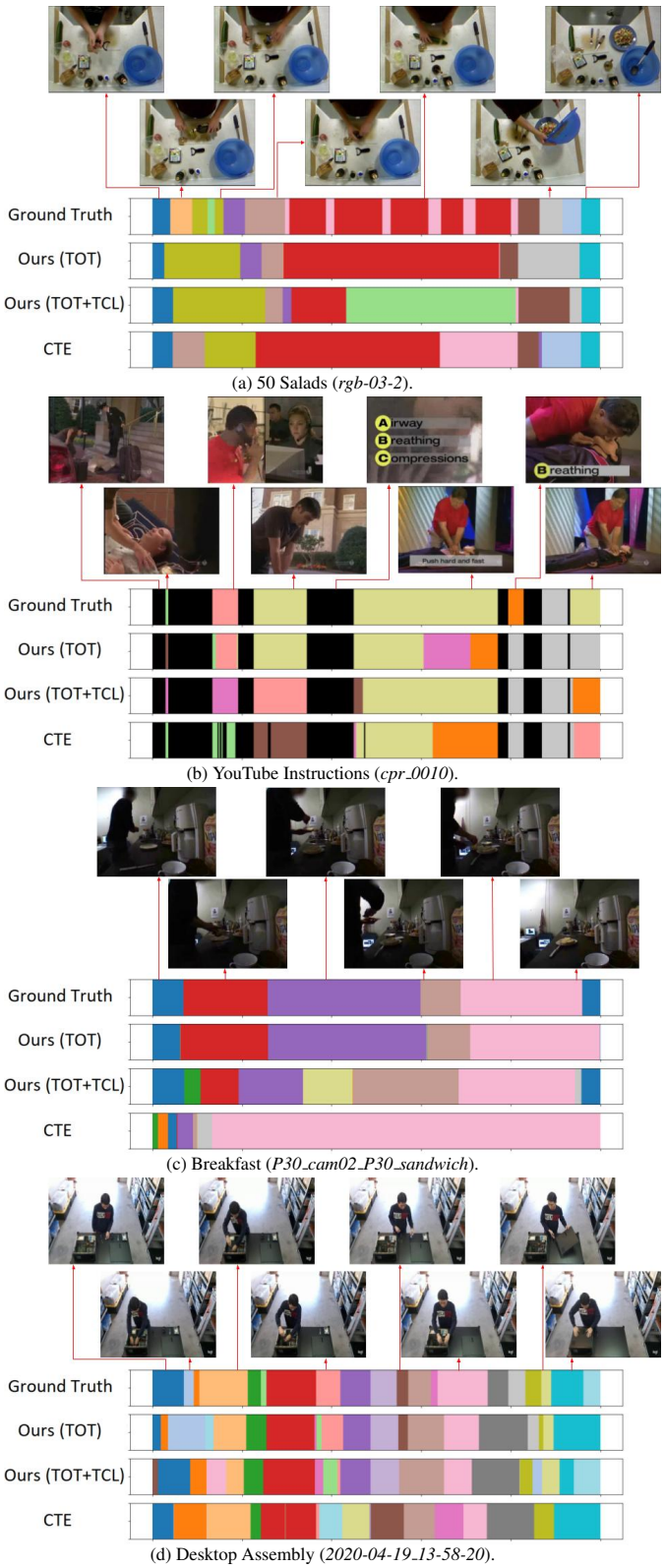
2. Qualitative Results

Fig. S1 shows some qualitative results on 50 Salads, YouTube Instructions, Breakfast, and Desktop Assembly datasets. Overall, the results of TOT and TOT+TCL are closer to the ground truth than those of CTE [6].

3. Implementation Details

Encoder Network. As mentioned in Sec. 4 of the main paper, we employ a two-layer fully-connected encoder network on top of the pre-computed features. Each fully-connected layer is followed by the *sigmoid* activation function. The dimensions of the output features are 30, 40 and 200 respectively for 50 Salads, Breakfast, and YouTube Instructions datasets.

Frame Sampling. As we mention in Sec. 3.2 of the main text, our temporal optimal transport module assumes a fixed order of the prototypes, and assigns early frames to early



(a) 50 Salads (*rgb-03-2*).

(b) YouTube Instructions (*cpr_0010*).

(c) Breakfast (*P30_cam02_P30_sandwich*).

(d) Desktop Assembly (*2020-04-19_13-58-20*).

Figure S1. Qualitative Segmentation Results on all the datasets.

prototypes and later frames to later prototypes. To implement the above, we sample frames from a video such that i) the sampled frames are temporally ordered and ii) the sampled frames spread over the entire video duration. In particular, we first divide the video into N bins of equal lengths. We then sample one anchor frame $\boldsymbol{z}_i$ from the i-th bin with $i \in \{1, 2, ..., N\}$. For the temporal coherence loss presented in Sec. 3.1 of the main paper, we sample a "positive" frame $\boldsymbol{z}_i^+$ for each anchor frame $\boldsymbol{z}_i$, i.e., $\boldsymbol{z}_i^+$ is inside a temporal window of λ from $\boldsymbol{z}_i$. Further, we consider all $\boldsymbol{z}_j^+$ with $j \neq i$ as "negative" frames for $\boldsymbol{z}_i$.

Background Class on Breakfast. The "SIL" action class in the Breakfast dataset corresponds to both background frames occurring at the start and at the end of the videos. However, the background frames at the start of the videos are visually and temporally different from those at the end of the videos. Therefore, following the 50 Salads dataset, we break the starting background frames and the ending background frames into 2 separate action classes (i.e., "action_start" and "action_end"). For a fair comparison, we have also evaluated CTE [6] with the above background label splitting, however that leads to performance drops on both F1-Score and MOF metrics. In particular, CTE with background label splitting obtains 22.7 F1-Score and 41.5% MOF, whereas CTE without background label splitting (in Tab. 5 of the main text) achieves 26.4 F1-Score and 41.8% MOF.

Adding Entropy Regularization to Eq. 9. The entropy regularization in Eq. 5 ensures cluster assignments are smoothly spread out among clusters but does not consider temporal positions of frames. The KL divergence in Eq. 9 takes both factors into account by considering temporal positions of frames and imposing a smooth prior distribution (Eq. 8) on cluster assignments. We did try adding the entropy regularization to Eq. 9 but did not get better results (for TOT on 50 Salads - *Eval* granularity, we obtained 46.2% vs. 47.4% in Tab. 3). Thus, we did not include the entropy regularization term in Eq. 9.

Hyperparameter Settings. The network is trained by using the ADAM optimizer [5] at a learning rate of 10^{-3} and a weight decay of 10^{-4}. We freeze the gradients for the prototypes during the first few iterations for better convergence [1]. For the three public datasets, we set τ to 0.1, λ to 30, and α to 1.0. Further, the number of Sinkhorn-Knopp iterations is fixed to 3 and each mini-batch contains sampled frames from 2 videos. Tabs. S1 and S2 present the hyperparameter settings for TOT and TOT+TCL respectively on the three public datasets, including 50 Salads, YouTube Instructions, and Breakfast.

Computing Resources. Our experiments are conducted with a single Nvidia V100 GPU on Microsoft Azure.

4. Desktop Assembly Dataset Details

Our Desktop Assembly dataset includes 76 videos of different actors assembling a desktop computer from its parts. The desktop assembly activity consists of 22 action classes and 1 background class, amounting to a total of 23 action classes. The actions are "picking up chip", "placing chip on motherboard", "closing cover", "picking up screw and screw driver", "tightening screw", "plugging stick in", "picking up fan", "placing fan on motherboard", "tightening screw A", "tightening screw B", "tightening screw C", "tightening screw D", "putting screw driver down", "connecting wire to motherboard", "picking up RAM", "installing RAM", "locking RAM", "picking up disk", "installing disk", "connecting wire A to motherboard", "connecting wire B to motherboard", "closing lid", and "background". The activity is performed by 4 different actors with various appearances, speeds, and viewpoints. We downsample the videos to 10 frames per second, resulting in a total of 59,165 frames for the entire dataset. We use ResNet-18 [4] pre-trained on ImageNet to obtain pre-computed features which are used as input for all methods. The original videos, pre-computed features, and action class labels are available at https://bit.ly/3JKm0JP. We note that the action class labels are only used during evaluation. Our hyperparameter settings for TOT and TOT+TCL on our Desktop Assembly dataset are presented in Tab. S3.

5. Societal Impacts

Our approach enables learning video recognition models without requiring action labels. It would positively impact the problems of worker training and assistance, where models automatically built from video datasets of expert demonstrations in diverse domains, e.g., factory work and medical surgery, could be used to provide training and guidance to new workers. Similarly, there exist problems such as surgery standardization, where operation room video datasets could be processed with approaches such as ours to improve the standard of care for patients globally. On the other hand, video understanding algorithms could generally be used in surveillance applications, where they improve security and productivity at the cost of privacy.

References

[1] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Neural Information Processing Systems*, 2020. 3

[2] Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *Proceedings of the IEEE international conference on computer vision*, pages 4086–4093, 2015. 1

Hyperparameter	Value
Rho (ρ)	0.07 (**E**), 0.08 (**M**), 0.08 (**Y**), 0.05 (**B**)
Sigma (σ)	2.5 (**E**), 2.0 (**M**), 1.25 (**Y**), 1.0 (**B**)
Mini-batch size	512
Temperature (τ)	0.1
Number of Sinkhorn-Knopp iterations	3
Learning rate	10^{-3}
Weight decay	10^{-4}
Number of videos per mini-batch	2

Table S1. Hyperparameter settings for TOT on the three public datasets, including 50 Salads, YouTube Instructions, and Breakfast. **E** denotes 50 Salads (*Eval* granularity), **M** denotes 50 Salads (*Mid* granularity), **Y** denotes YouTube Instructions, and **B** denotes Breakfast.

Hyperparameter	Value
Rho (ρ)	0.08 (**E**), 0.07 (**M**), 0.07 (**Y**), 0.04 (**B**)
Sigma (σ)	2.5 (**E**), 1.75 (**M**), 3.0 (**Y**), 0.75 (**B**)
Mini-batch size	512
Temperature (τ)	0.1
Window size (λ)	30
Alpha (α)	1.0
Number of Sinkhorn-Knopp iterations	3
Learning rate	10^{-3}
Weight decay	10^{-4}
Number of videos per mini-batch	2

Table S2. Hyperparameter settings for TOT+TCL on the three public datasets, including 50 Salads, YouTube Instructions, and Breakfast. **E** denotes 50 Salads (*Eval* granularity), **M** denotes 50 Salads (*Mid* granularity), **Y** denotes YouTube Instructions, and **B** denotes Breakfast.

Hyperparameter	Value
Rho (ρ)	0.07
Sigma (σ)	2.0
Mini-batch size	512
Temperature (τ)	0.1
Window size (λ)	30
Alpha (α)	1.0
Number of Sinkhorn-Knopp iterations	3
Learning rate	10^{-3}
Weight decay	10^{-4}
Number of videos per mini-batch	2

Table S3. Hyperparameter settings for TOT and TOT+TCL on our Desktop Assembly dataset. Window size (λ) and Alpha (α) are only used in TOT+TCL.

[3] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006. 1

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[6] 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. 2, 3

[7] Jun Li and Sinisa Todorovic. Action shuffle alternating learning for unsupervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[8] Hossein Mobahi, Ronan Collobert, and Jason Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 737–744, 2009. 1

[9] 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. 2

[10] Sirnam Swetha, Hilde Kuehne, Yogesh S Rawat, and Mubarak Shah. Unsupervised discriminative embedding for sub-action learning in complex activities. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2588–2592. IEEE, 2021. 2

[11] 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. 2