

# Temporal Rate Reduction Clustering for Human Motion Segmentation

## Supplementary Material

### A. Experimental Supplementary Material

#### A.1. Datasets Description

**Weizmann action dataset (Weiz).** The Weizmann dataset [17] contains 90 motion sequences, with 9 individuals each completing 10 motions, e.g., running, jumping, skipping, waving and bending. The resolution of video is  $180 \times 144$  pixels with 50 FPS.

**Keck gesture dataset (Keck).** The Keck dataset [21] contains 56 action sequences, with 4 individuals each performing 14 motions derived from military hand signals, e.g., turning left, turning right, starting, and speeding up. The resolution of video is  $640 \times 480$  pixels with 15 FPS.

**UT interaction dataset (UT).** The UT dataset [41] contains 10 video sequences, each of which consists of 2 people completing 6 different motions, e.g., shaking hands, hugging, pointing, and kicking. The resolution of video is  $720 \times 480$  pixels with 30 FPS.

**Multi-model Action Detection dataset (MAD).** The MAD dataset [19] contains 40 video sequences (20 people, 2 videos each) with 35 motions in each video. The resolution of video is  $320 \times 240$  pixels with 30 FPS. The dataset gives both depth data and skeleton data.

**UCF-11 YouTube action dataset (YouTube).** The YouTube dataset [31] contains 1168 video sequences with 11 motions, e.g., biking, diving, and golf swinging. The resolution of video is  $320 \times 240$  pixels with 30 FPS. Specifically, the human motions in the YouTube dataset are partially associated with objects such as horses, bikes, or dogs.

To have a fair comparison with the baselines, we cut down the number of human motions of Keck, MAD and YouTube datasets to 10. For Keck, Weiz and YouTube datasets in which each video captures only one human motion, we concatenate the original videos and conduct experiments on the resulting videos.

#### A.2. List of Hyper-Parameters

The hyper-parameters of training  $\text{TR}^2\text{C}$  are summarized in Table A.1. We choose the same hidden dimension  $d_{pre}$ , output dimension  $d$ , window size  $s$ , coding precision  $\epsilon$ , and learning rate  $\eta$  for all the experiments and tune the weights  $\lambda_1$  and  $\lambda_2$  for each dataset. For training on CLIP features, we decrease the number of training iterations from 500 to 100 due to the faster convergence, while keeping all the other hyper-parameters unchanged.

Table A.1. Detailed hyper-parameters configuration for training  $\text{TR}^2\text{C}$  with different feature extractors.

Features	Dataset	$d_{pre}$	$d$	$T$	$\lambda_1$	$\lambda_2$	$s$	$\epsilon$	$\eta$
HoG	Weiz	512	64	500	0.1	12	2	0.1	$5 \times 10^{-3}$
	Keck	512	64	500	0.1	10	2	0.1	$5 \times 10^{-3}$
	UT	512	64	500	0.1	10	2	0.1	$5 \times 10^{-3}$
	MAD	512	64	500	0.15	15	2	0.1	$5 \times 10^{-3}$
VGG	YouTube	512	64	500	1	2	2	0.1	$5 \times 10^{-3}$
CLIP	Weiz	512	64	100	0.1	12	2	0.1	$5 \times 10^{-3}$
	Keck	512	64	100	0.1	10	2	0.1	$5 \times 10^{-3}$
	YouTube	512	64	100	1	2	2	0.1	$5 \times 10^{-3}$

#### A.3. Visualization of Representations by GCTSC and $\text{TR}^2\text{C}$

In the main text, we have visualized the representations from different motions by different colors to demonstrate the union-of-orthogonal-subspaces distribution of learned representations. To further demonstrate the temporal continuity of learned representations, we visualize the data points (i.e., the feature vectors of frames in video) with a continuously varying colormap. As illustrated in Figure A.1 (in the first row), while the temporal consistency is preserved, the distribution of the representations learned by  $\text{TR}^2\text{C}$  are “compressed” in a structured way; whereas the learned representations by GCTSC (in

the second row) do preserve the temporal continuity very well, but lack of specific structures to facilitate the task of motion segmentation.

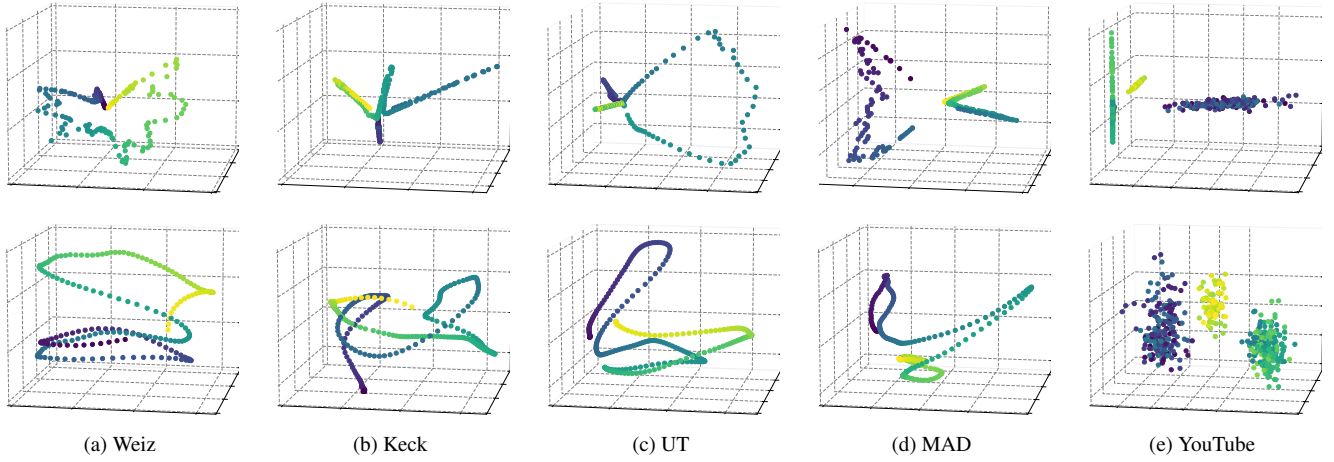


Figure A.1. **PCA visualization of learned representations.** First row: representations learned by  $TR^2C$ . Second row: representations learned by GCTSC. We conduct experiments on the first sequence of each dataset.

#### A.4. Clustering Performance Evaluation on Different Representations

To further validate the effectiveness of  $TR^2C$ , we use the HoG features, the representations learned by GCTSC and  $TR^2C$  as the input and evaluate the performance of different methods, including Spectral Clustering (SC), Elastic Net Subspace Clustering (EnSC), TSC and GCTSC. As shown in Figure A.2, the clustering performance of representations from  $TR^2C$  consistently surpasses that of HoG features, regardless of the datasets and clustering methods used. The performance gap is notably larger when clustering with SC and EnSC, as these classical clustering approaches overlook the temporal consistency prior of HMS. In contrast, the performance improvements in TSC and GCTSC, which incorporate temporal consistency regularizers, are largely driven by the union-of-orthogonal-subspaces distribution learned by  $TR^2C$ . Notably, the representations learned by GCTSC also achieve satisfying clustering performance, though they do not outperform  $TR^2C$ , except for clustering with SC on datasets Weizmann and MAD. It is surprising that the accuracy yields by  $\Gamma$  of  $TR^2C$  even outperforms “ $TR^2C$  features+GCTSC” on all the datasets except for the MAD, implying that the reparameterized affinity matrix is better at capturing the subspace membership.

#### A.5. Ablation Study

We report the ablation study results of all the benchmarks in Table A.2. The performances are averaged across all the sequences of each dataset. As analyzed in the main text, each term in  $TR^2C$  is indispensable for the learning of temporally consistent representations that align with a union of orthogonal subspaces.

Table A.2. **Ablation study.** We report the average performance of all the sequences.

$\mathcal{L}_p$	Loss		Weiz		Keck		UT		MAD		YouTube	
	$\mathcal{L}_{\tilde{p}^c}$	$\mathcal{L}_r$	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
✓	✓		37.30	45.86	47.29	49.78	45.79	35.30	30.27	29.40	94.82	97.30
	✓	✓	53.14	61.51	47.91	51.39	63.13	59.51	50.54	53.23	96.07	97.77
✓		✓	64.68	74.67	58.60	65.21	65.67	66.09	64.91	72.37	48.16	53.36
✓			41.21	44.57	44.01	41.46	46.80	37.49	28.00	22.97	58.87	54.08
	✓		56.03	64.19	47.50	52.11	76.39	72.41	43.23	43.11	90.15	91.79
		✓	52.59	60.33	48.35	50.87	62.13	58.29	50.54	53.13	96.01	97.52
✓	✓	✓	<b>94.07</b>	<b>96.08</b>	<b>86.78</b>	<b>86.93</b>	<b>94.05</b>	<b>92.34</b>	<b>83.99</b>	<b>87.32</b>	<b>96.40</b>	<b>98.50</b>

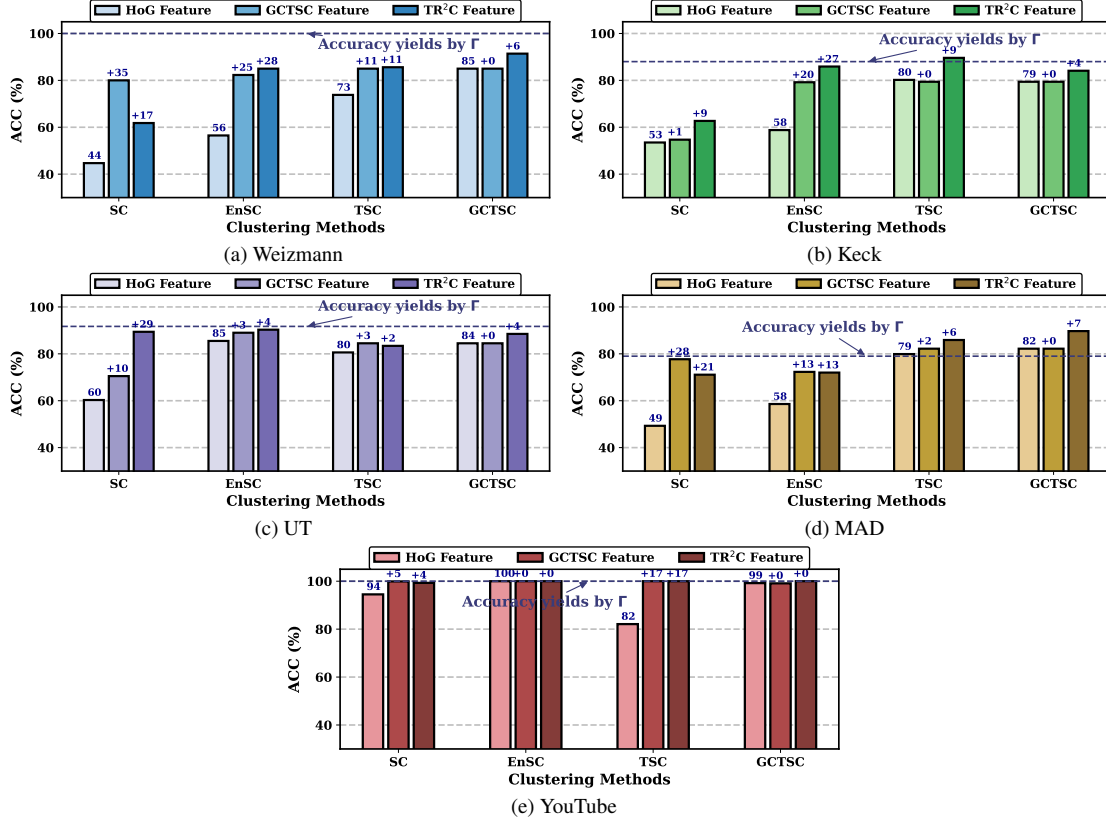


Figure A.2. Clustering performance evaluation on different representations.

## A.6. Complexity Analysis

We analyze the time complexity of  $\log \det(\cdot)$  operation, as it is the most computationally intensive component in  $\text{TR}^2\text{C}$ . By the commutative property:  $\log \det(\mathbf{I} + \mathbf{Z}\mathbf{Z}^\top) = \log \det(\mathbf{I} + \mathbf{Z}^\top\mathbf{Z})$  (see [34]), we reduce the matrix size involved in  $\log \det(\cdot)$  from  $N \times N$  to  $d \times d$ , which significantly improves both time and memory efficiency, especially when  $d \ll N$ . Since that  $-\mathcal{L}_\rho + \mathcal{L}_{\bar{\rho}^c}$  requires computing  $\log \det(\cdot)$  for  $N + 1$  times, the complexity of our loss becomes  $\mathcal{O}(Nd^3)$ , which can be further accelerated with GPU support. We report the time cost (ms/iter) of  $\text{TR}^2\text{C}$  with varying  $N$  on HoG features ( $d = 324$ ) in Table A.3. As can be seen, both the time and memory cost of  $\text{TR}^2\text{C}$  are significantly reduced by exploiting the commutative property of  $\log \det(\cdot)$  operation.

Table A.3. Time cost (ms/iter) with varying  $N$  on HoG features. “OOM” refers to out-of-memory.

$N$	200	400	600	800	1000	2000	3000	4000	Complexity
w/o Commutation	33.2	97.1	229.1	546.8	1039.3	OOM	OOM	OOM	$\mathcal{O}(N^4)$
Our $\text{TR}^2\text{C}$	16.1	17.7	21.3	23.6	28.0	53.9	105.2	162.9	$\mathcal{O}(Nd^3)$

## A.7. Learning Curves

We plot the learning curves with respect to  $\mathcal{L}_\rho - \lambda_1 \mathcal{L}_{\bar{\rho}^c}$ ,  $\mathcal{L}_\rho$ ,  $\mathcal{L}_{\bar{\rho}^c}$ ,  $\mathcal{L}_r$  and the clustering performance in Figure A.3. As illustrated, the gap between  $\mathcal{L}_\rho$  and  $\mathcal{L}_{\bar{\rho}^c}$  increases rapidly as the  $\mathcal{L}_\rho - \lambda_1 \mathcal{L}_{\bar{\rho}^c}$  decreases, encouraging the UoS structure of learned representations. The  $\mathcal{L}_r$  decreases, promoting the temporal continuity of learned representations. Consequently, the clustering results gradually converge to state-of-the-art performances.

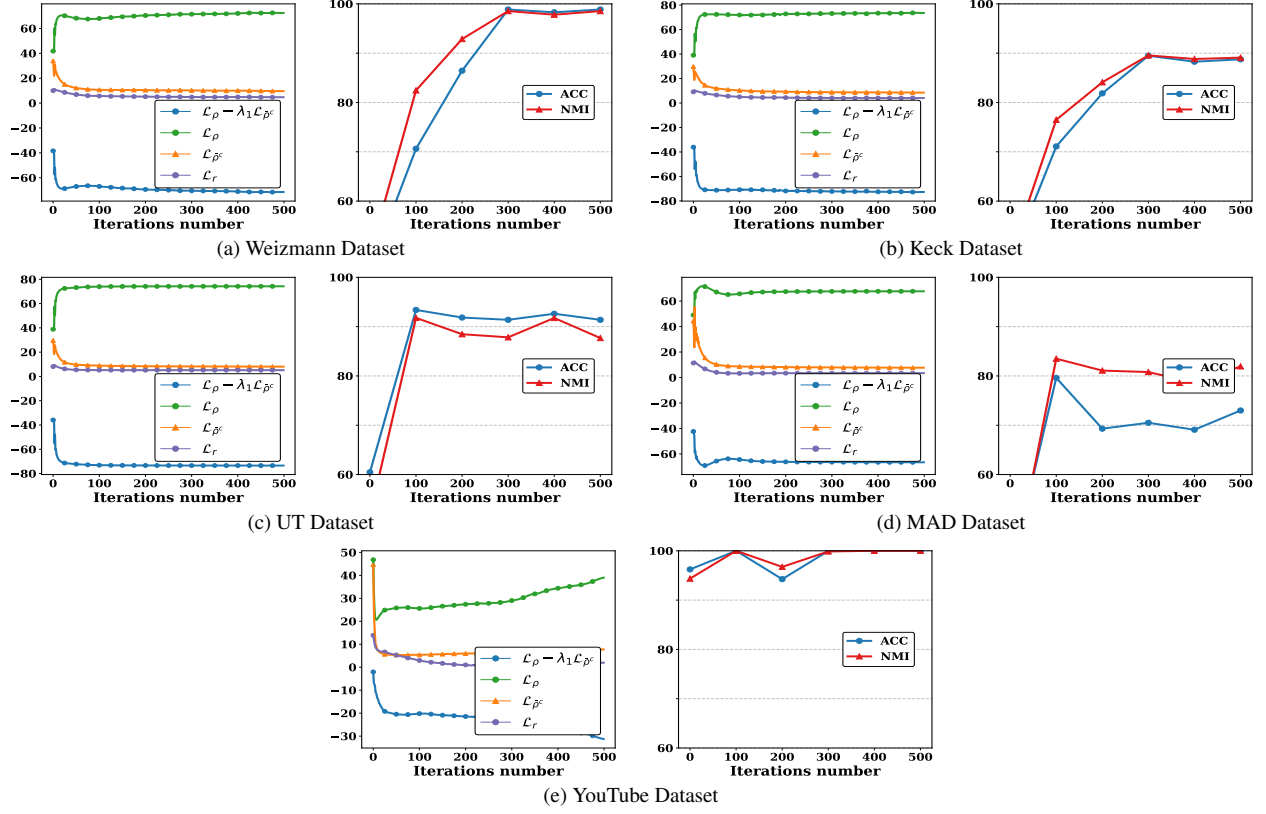


Figure A.3. Learning curves of the TR<sup>2</sup>C framework on HoG features.

## A.8. Segmentation Results Visualization

To qualitatively demonstrate the effectiveness of TR<sup>2</sup>C, we visualize the video segmentation results along with the ground-truth labels for the first three sequences on the five benchmark datasets. Notably, our unsupervised TR<sup>2</sup>C produces segmentation results that closely match the manually annotated ground-truth labels on the Weizmann, UT, and YouTube datasets. For the Keck and MAD datasets, segmentation errors primarily occur in frames capturing transitions between different human motions. For instance, in the Keck dataset, these frames often show individuals adjusting their standing positions, making it inherently difficult to determine whether they belong to the preceding or the subsequent motion.

## A.9. Compared to CLIP Zero-Shot

Next, using the pretrained CLIP model, we explore the performance of zero-shot learning in the HMS task. For the Weizmann dataset, we first convert the ground-truth labels of the dataset into textual descriptions of each motion. For instance, the motion “Wave1” is described as “A photo of people waving one hand.” (see Table A.5 for all the descriptions). Then, we extract text embeddings for all the descriptions using a pretrained text encoder of CLIP. For each frame in the dataset, we match its image embedding to the text embedding with the highest cosine similarity and assign the corresponding description as the zero-shot classification result for that frame.

As shown in Figure A.5, the classification accuracy for “Walk” and “Wave1” is 99.89% and 97.40%, respectively, making them two of the best-performing classes. However, the overall classification accuracy is only 29.14%, which is significantly lower than the performance of TR<sup>2</sup>C+CLIP (96.21%). Notably, 63.31% of frames are misclassified as “Walk” while no samples from the “Jack”, “Jump”, “PJump” and “Side” classes are correctly identified. If we reduce the difficulty by using coarse labels consisting only of “bend”, “jump”, “run”, “walk” and “wave” (Table A.4), the accuracy of zero-shot classification for HMS is 39.87%, which is still significantly lower than the performance of TR<sup>2</sup>C+CLIP (96.21%).

These results demonstrate that vanilla zero-shot classification is not suitable for HMS, which due to the fact that zero-shot learning classifies frames individually, failing to capture in-context information. In contrast, TR<sup>2</sup>C+CLIP succeeds by learning temporally consistent representations that align with a union of orthogonal subspaces.

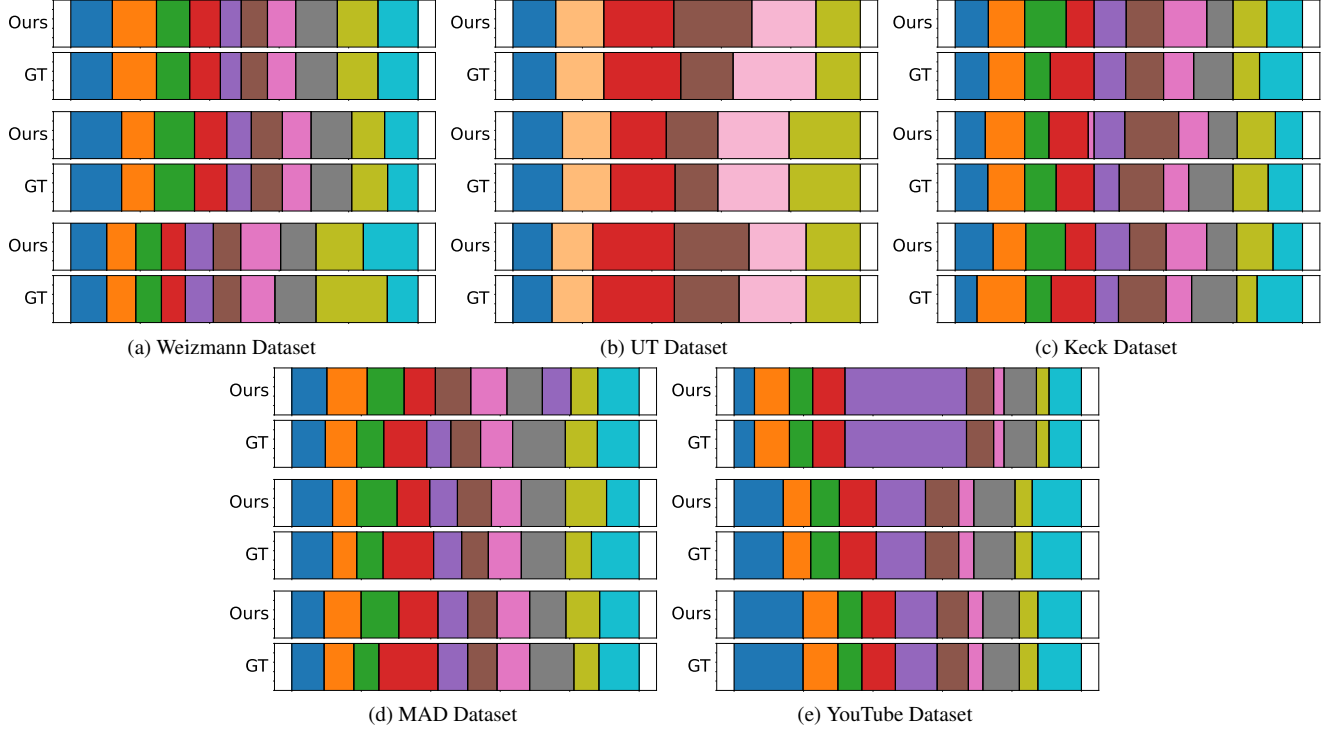


Figure A.4. Segmentation results visualization.

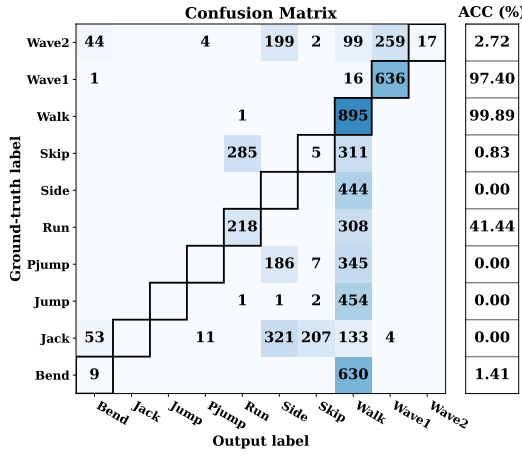


Figure A.5. Confusion matrix of zero-shot classification result for HMS task.

Table A.4. Zero-shot classification with coarse label for HMS task. The coarse ground-truth label is marked in blue.

GT	Bend	Jump	Target Run	Walk	Wave	ACC (%)
Bend	10			629		1.56
Jack	83	270	7	327	42	37.04
Jump				458		0.00
Pjump		107		431		19.89
Run			218	308		41.44
Side				444		0.00
Skip			290	311		0.00
Walk			1	895		99.89
Wave1	4			44	605	92.65
Wave2	77			217	330	52.88

## A.10. Experimental Details for Temporal Action Segmentation

**Datasets description.** The Breakfast dataset [22] consists of 1,712 videos capturing 52 participants performing 10 activities, including making friedegg, sandwich, pancake, *et al.* The YouTube Instructional dataset [2] consists of 150 videos with 5 activities capturing complex interactions between people and objects, including changing tire, making coffee, repotting, *et al.* The 50 Salads dataset [47] consists of 50 videos capturing people preparing mixed salads from a top-down perspective. We follow the baselines for the feature extractor selection of each dataset. For the Breakfast and 50 Salads dataset, we use the Improved Dense Trajectory (IDT) [49] features provided by [23]; and for YouTube Instructional dataset, we use a concatenation of HOF descriptors [24] and VGG features [44].

Table A.5. **Textual description for zero-shot classification of Weizmann dataset.**











#	Label	Icon	Textual Description	#	Label	Icon	Textual Description
1	Bend		A photo of people bending.	6	Side		A photo of people side jumping.
2	Jack		A photo of people jumping jacks.	7	Skip		A photo of people skipping jump.
3	Jump		A photo of people jumping.	8	Walk		A photo of people walking.
4	Pjump		A photo of people jumping in place.	9	Wave1		A photo of people waving one hand.
5	Run		A photo of people running.	10	Wave2		A photo of people waving two hands.

Table A.6. **Hyper-parameters configuration for training TR<sup>2</sup>C on temporal action segmentation benchmark datasets.**

Dataset	$d_{pre}$	$d$	$T$	$\lambda_1$	$\lambda_2$	$s$	$\epsilon$	$\eta$
Breakfast	64	64	100	0.05	12	2	0.1	$10^{-3}$
YouTube Instr.	512	64	500	0.05	20	2	0.05	$10^{-2}$
50 Salads	256	64	500	0.05	15	2	0.05	$10^{-2}$

**Experimental details.** A significant distinction of the TAS benchmark datasets compared to that of HMS is that it contains a higher number of frames per video (e.g., the average number of frames per video of 50 Salads is 11,788). To address this discrepancy while maintaining computational tractability, we down-sample each video before training TR<sup>2</sup>C, then up-sample the segmentation result back to the original number of frames. Commonly used evaluation metrics, namely, Mean over Frames (MoF), F1-score, and Intersection over Union (IoU) are computed following the baselines. The architecture of neural networks remains consistent with the experiments on HMS. The hyper-parameters configuration of training TR<sup>2</sup>C is listed in Table A.6. When applying the state-of-the-art TAS methods to HMS datasets, we report the best results after tuning hyper-parameters which are picked from the Table A.7.

Table A.7. **Hyper-parameters tuning for temporal action segmentation methods on HMS datasets.**

Method	Hyper-parameters for Tuning
TWF [42]	N/A (Automatic Clustering, no parameter to tune)
ASOT [56]	$\alpha \in \{0.2, 0.5\}, r \in \{0.02, 0.04, 0.06, 0.08, 0.1\}, \rho \in \{0.3, 0.5, 0.7\}, \lambda \in \{0.08, 0.11, 0.14, 0.17, 0.2\}$
HVQ [46]	$\alpha \in \{1, 2, 3, 4\}, \lambda_{rec} \in \{0.0005, 0.002, 0.1\}$