Supplementary Material for "Action Segmentation with Joint Self-Supervised Temporal Domain Adaptation"

Min-Hung Chen[1]* Baopu Li[2] Yingze Bao[2] Ghassan AlRegib[1] Zsolt Kira[1]
[1]Georgia Institute of Technology [2]Baidu USA

In the supplementary material, we would like to show more details about the technical approach, implementation, and experiments.

1. Technical Approach Details

1.1. Domain Attentive Temporal Pooling (DATP)

Temporal pooling is one of the most common methods to aggregate frame-level features into video-level features for each video. However, not all the frame-level features contribute the same to the overall domain discrepancy. Therefore, inspired by [2, 3], we assign larger attention weights to the features which have larger domain discrepancy so that we can focus more on aligning those features, achieving more effective domain adaptation.

More specifically, we utilize the entropy criterion to generate the domain attention value for each frame-level feature f_j as below:

$$\hat{w}_j = 1 - H(\hat{d}_j) \tag{1}$$

where $\hat{d}_j$ is the output from the learned domain classifier G_{ld} used in local SSTDA. $H(p) = -\sum_k p_k \cdot \log(p_k)$ is the entropy function to measure uncertainty. $\hat{w}_j$ increases when $H(\hat{d}_j)$ decreases, which means the domains can be distinguished well. We also add a residual connection for more stable optimization. Finally, we aggregate the attended frame-level features with temporal pooling to generate the video-level feature v, which is noted as *Domain Attentive Temporal Pooling (DATP)*, as illustrated in the left part of Figure 1 and can be expressed as:

$$v = \frac{1}{T'}\sum_{j=1}^{T'}(\hat{w}_j + 1) \cdot f_j \tag{2}$$

where $+1$ refers to the residual connection, and $\hat{w}_j + 1$ is equal to w_j in the main paper. T' is the number of frames used to generate a video-level feature.

Local SSTDA is necessary to calculate the attention weights for DATP. Without this mechanisms, frames will be aggregated in the same way as *temporal pooling* without cross-domain consideration, which is already demonstrated sub-optimal for cross-domain video tasks [2, 3].

*Work done during an internship at Baidu USA

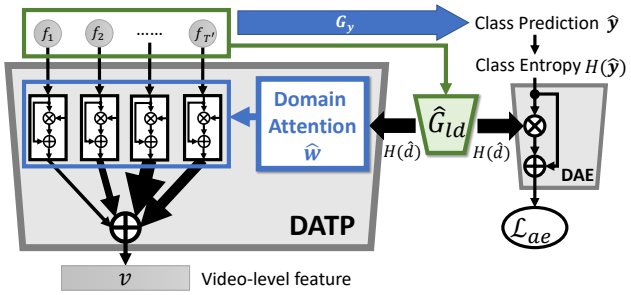


Figure 1: The details of DATP (*left*) and DAE (*right*). Both modules take the domain entropy $H(\hat{d})$, which is calculated from the domain prediction $\hat{d}$, to calculate the attention weights. With the residual connection, DATP attends to the frame-level features for aggregating into the final video-level feature v (arrow thickness represents assigned attention values), and DAE attends to the class entropy $H(\hat{\boldsymbol{y}})$ to obtain the attentive entropy loss $\mathcal{L}_{ae}$.

1.2. Domain Attentive Entropy (DAE)

Minimum entropy regularization is a common strategy to perform more refined classifier adaptation. However, we only want to minimize class entropy for the frames that are similar across domains. Therefore, inspired by [14], we attend to the frames which have low domain discrepancy, corresponding to high domain entropy $H(\hat{d}_j)$. More specifically, we adopt the *Domain Attentive Entropy (DAE)* module to calculate the attentive entropy loss $\mathcal{L}_{ae}$, which can be expressed as follows:

$$\mathcal{L}_{ae} = \frac{1}{T}\sum_{j=1}^{T}(H(\hat{d}_j) + 1) \cdot H(\hat{y}_j) \tag{3}$$

where $\hat{d}$ and $\hat{y}$ is the output of $\hat{G}_{ld}$ and G_y, respectively. T is the total frame number of a video. We also apply the residual connection for stability, as shown in the right part of Figure 1.

	GTEA	50Salads	Breakfast
subject #	4	25	52
class #	11	17	48
video #	28	50	1712
avg. length (min.)	1	6.4	2.7
avg. action #/video	20	20	6
cross-validation	4-fold	5-fold	4-fold
leave-#-subject-out	1	5	13

Table 1: The statistics of action segmentation datasets.

1.3. Full Architecture

Our method is built upon the state-of-the-art action segmentation model, MS-TCN [4], which takes input frame-level feature representations and generates the corresponding output frame-level class predictions by four stages of SS-TCN. In our implementation, we convert the second and third stages into *Domain Adaptive TCN (DA-TCN)* by integrating each SS-TCN with the following three parts: 1) $\hat{G}_{ld}$ (for *binary domain prediction*), 2) DATP and $\hat{G}_{gd}$ (for *sequential domain prediction*), and 3) DAE, bringing three corresponding loss functions, $\mathcal{L}_{ld}$, $\mathcal{L}_{gd}$ and $\mathcal{L}_{ae}$, respectively, as illustrated in Figure 2. The final loss function can be formulated as below:

$$\mathcal{L} = \sum_{s=1}^{4} \mathcal{L}_{y_{(s)}} - \sum_{s=2}^{3} (\beta_l \mathcal{L}_{ld_{(s)}} + \beta_g \mathcal{L}_{gd_{(s)}} - \mu \mathcal{L}_{ae_{(s)}}) \quad (4)$$

where β_l, β_g and μ are the weights for $\mathcal{L}_{ld}$, $\mathcal{L}_{gd}$ and $\mathcal{L}_{ae}$, respectively, obtained by the methods described in Section 2.2. s is the stage index in MS-TCN.

2. Experiments

2.1. Datasets and Evaluation Metrics

The detailed statistics and the evaluation protocols of the three datasets are listed in Table 1. We follow [7] to use the following three metrics for evaluation:

1. *Frame-wise accuracy (Acc)*: Acc is one of the most typical evaluation metrics for action segmentation, but it does not consider the temporal dependencies of the prediction, causing the inconsistency between qualitative assessment and frame-wise accuracy. Besides, long action classes have higher impact on this metric than shorter action classes, making it not able to reflect over-segmentation errors.

2. *Segmental edit score (Edit)*: The edit score penalizes over-segmentation errors by measuring the ordering of predicted action segments independent of slight temporal shifts.

3. *Segmental F1 score at the IoU threshold $k\%$ (F1@k)*: F1@k also penalizes over-segmentation errors while ignoring minor temporal shifts between the predictions and ground truth. The scores are determined by the total number of actions but do not depend on the duration of each action instance, which is similar to mean average precision (mAP) with intersection-over-union (IoU) overlap criteria. F1@k becomes popular recently since it better reflects the qualitative results.

2.2. Implementation and Optimization

Our implementation is based on the PyTorch [10, 13] framework. We extract I3D [1] features for the video frames and use these features as inputs to our model. The video frame rates are the same as [4]. For GTEA and Breakfast datasets we use a video temporal resolution of 15 frames per second (fps), while for 50Salads we downsampled the features from 30 fps to 15 fps to be consistent with the other datasets. For fair comparison, we adopt the same architecture design choices of MS-TCN [4] as our baseline model. The whole model consists of four stages where each stage contains ten dilated convolution layers. We set the number of filters to 64 in all the layers of the model and the filter size is 3. For optimization, we utilize the Adam optimizer and a batch size equal to 1, following the official implementation of MS-TCN [4]. Since the target data size is smaller than the source data, each target data is loaded randomly multiple times in each epoch during training. For the weighting of loss functions, we follow the common strategy as [5, 6] to gradually increase β_l and β_g from 0 to 1. The weighting α for smoothness loss is 0.15 as in [4] and μ is chosen as 1×10^{-2} via the grid-search.

2.3. Less Training Labeled Data

To investigate the potential to train with a fewer number of labeled frames using SSTDA, we drop labeled frames from source domains with uniform sampling for training, and evaluate on the same length of validation data. Our experiment on the 50Salads dataset shows that by integrating with SSTDA, the performance does not drop significantly with the decrease in labeled training data, indicating the alleviation of reliance on labeled training data. Finally, only *65%* of labeled training data are required to achieve comparable performance with MS-TCN, as shown in Table 2. We then evaluate the proposed SSTDA on GTEA and Breakfast with the same percentage of labeled training data, and also get comparable or better performance.

Table 2 also indicates the results without additional labeled training data, which contain discriminative information that can directly boost the performance for action segmentation. The additional trained data are all unlabeled, so they cannot be directly trained with standard prediction loss. Therefore, we propose SSTDA to exploit unlabeled data to:

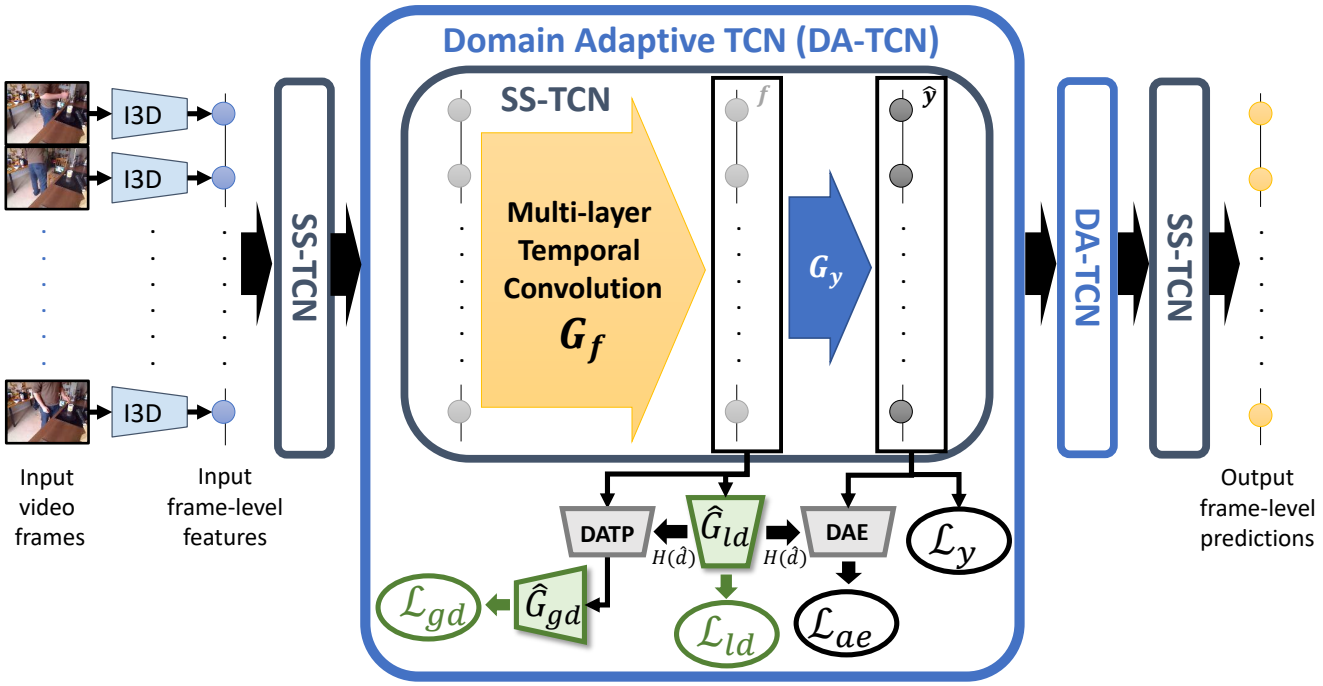


Figure 2: The overall architecture of the proposed SSTDA. By equipping the network with a local adversarial domain classifier $\hat{G}_{ld}$, a global adversarial domain classifier $\hat{G}_{gd}$, a domain attentive temporal pooling (DATP) module, and a domain attentive entropy (DAE) module, we convert a SS-TCN into a DA-TCN, and stack multiple SS-TCNs and DA-TCNs to build the final architecture. $\mathcal{L}_{ld}$ and $\mathcal{L}_{gd}$ is the local and global domain loss, respectively. $\mathcal{L}_y$ is the prediction loss and $\mathcal{L}_{ae}$ is the attentive entropy loss. The domain entropy $H(\hat{d})$ is used to calculate the attention weights for DATP and DAE. An adversarial domain classifier $\hat{G}$ refers to a domain classifier G equipped with a gradient reversal layer (GRL).

50Salads	m%	F1@{10, 25, 50}			Edit	Acc
SSTDA	100%	83.0	81.5	73.8	75.8	83.2
	95%	81.6	80.0	73.1	75.6	83.2
	85%	81.0	78.9	70.9	73.8	82.1
	75%	78.9	76.5	68.6	71.7	81.1
	65%	**77.7**	**75.0**	**66.2**	**69.3**	**80.7**
MS-TCN	100%	75.4	73.4	65.2	68.9	82.1
GTEA	m%	F1@{10, 25, 50}			Edit	Acc
SSTDA	100%	90.0	89.1	78.0	86.2	79.8
	65%	**85.2**	**82.6**	**69.3**	**79.6**	**75.7**
MS-TCN	100%	86.5	83.6	71.9	81.3	76.5
Breakfast	m%	F1@{10, 25, 50}			Edit	Acc
SSTDA	100%	75.0	69.1	55.2	73.7	70.2
	65%	**69.3**	**62.9**	**49.4**	**69.0**	**65.8**
MS-TCN	100%	65.3	59.6	47.2	65.7	64.7

Table 2: The comparison of SSTDA trained with less labeled training data. m in the first row indicates the percentage of labeled training data used to train a model.

1) further improve the strong baseline, MS-TCN, without additional training labels, and 2) achieve comparable performance with this strong baseline using only 65% of labels for training.

2.4. Comparison with Other Approaches

We compare our proposed SSTDA with other approaches by integrating the same baseline architecture with other popular DA methods [6, 9, 11, 15, 12, 8] and a state-of-the-art video-based self-supervised approach [16]. For fair comparison, all the methods are integrated with the second and third stages, as our proposed SSTDA, where the single-stage integration methods are described as follows:

1. *DANN [6]*: We add one discriminator, which is the same as G_{ld}, equipped a gradient reversal layer (GRL) to the final frame-level features $\boldsymbol{f}$.

2. *JAN [9]*: We integrate Joint Maximum Mean Discrepancy (JMMD) to the final frame-level features $\boldsymbol{f}$ and the class prediction $\hat{\boldsymbol{y}}$.

3. *MADA [11]*: Instead of a single discriminator, we add multiple discriminators according to the class number to calculate the domain loss for each class. All the class-based domain losses are weighted with prediction probabilities and then summed up to obtain the final domain loss.

50Salads	F1@{10, 25, 50}			Edit
Source only (MS-TCN)	75.4	73.4	65.2	68.9
VCOP [16]	75.8	73.8	65.9	68.4
DANN [6]	79.2	77.8	70.3	72.0
JAN [9]	80.9	79.4	72.4	73.5
MADA [11]	79.6	77.4	70.0	72.4
MSTN [15]	79.3	77.6	71.5	72.1
MCD [12]	78.2	75.5	67.1	70.8
SWD [8]	78.2	76.2	67.4	71.6
SSTDA	**83.0**	**81.5**	**73.8**	**75.8**
Breakfast	F1@{10, 25, 50}			Edit
Source only (MS-TCN)	65.3	59.6	47.2	65.7
VCOP [16]	68.5	62.9	50.1	67.9
DANN [6]	72.8	67.8	55.1	71.7
JAN [9]	70.2	64.7	52.0	70.0
MADA [11]	71.0	65.4	52.8	71.2
MSTN [15]	69.6	63.6	51.5	69.2
MCD [12]	70.4	65.1	52.4	69.7
SWD [8]	68.6	63.2	50.6	69.1
SSTDA	**75.0**	**69.1**	**55.2**	**73.7**

Table 3: The comparison of different methods that can learn information from unlabeled target videos (on 50Salads and Breakfast). All the methods are integrated with the same baseline model MS-TCN for fair comparison.

4. *MSTN [15]*: We utilize pseudo-labels to cluster the data from the source and target domains, and calculate the class centroids for the source and target domain separately. Then we compute the semantic loss by calculating mean squared error (MSE) between the source and target centroids. The final loss contains the prediction loss, the semantic loss, and the domain loss as DANN [6].

5. *MCD [12]*: We apply another classifier G'_y and follow the adversarial training procedure of Maximum Classifier Discrepancy to iteratively optimize the generator (G_f in our case) and the classifier (G_y). The L1-distance is used as the discrepancy loss.

6. *SWD [8]*: The framework is similar to MCD, but we replace the L1-distance with the Wasserstein distance as the discrepancy loss.

7. *VCOP [16]*: We divide $\boldsymbol{f}$ into three segments and compute the segment-level features with temporal pooling. After temporal shuffling the segment-level features, pairwise features are computed and concatenated into the final feature representing the video clip order. The final features are then fed into a shallow classifier to predict the order.

The experimental results on 50Salads and Breakfast both indicate that our proposed SSTDA outperforms all these methods, as shown in Table 3.

The performance of the most recent video-based self-supervised learning method [16] on 50Salads and Breakfast also show that temporal shuffling *within single domain* without considering the relation across domains does not effectively benefit cross-domain action segmentation, resulting in even worse performance than other DA methods. Instead, our proposed self-supervised auxiliary tasks make predictions on cross-domain data, leading to cross-domain temporal relation reasoning instead of predicting within-domain temporal orders, achieving significant improvement in the performance of our main task, action segmentation.

3. Segmentation Visualization

Here we show more qualitative segmentation results from all three datasets to compare our methods with the baseline model, MS-TCN [4]. All the results (Figure 3 for *GTEA*, Figure 4 for *50Salads*, and Figure 5 for *Breakfast*) demonstrate that the improvement over the baseline by only local SSTDA is sometimes limited. For example, local SSTDA falsely detects the *pour* action in Figure 3b, falsely classifies *cheese*-related actions as *cucumber*-related actions in Figure 4b, and falsely detects the *stir milk* action in Figure 5b. However, by jointly aligning local and global temporal dynamics with SSTDA, the model is effectively adapted to the target domain, reducing the above mentioned incorrect predictions and achieving better segmentation.

References

[1] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

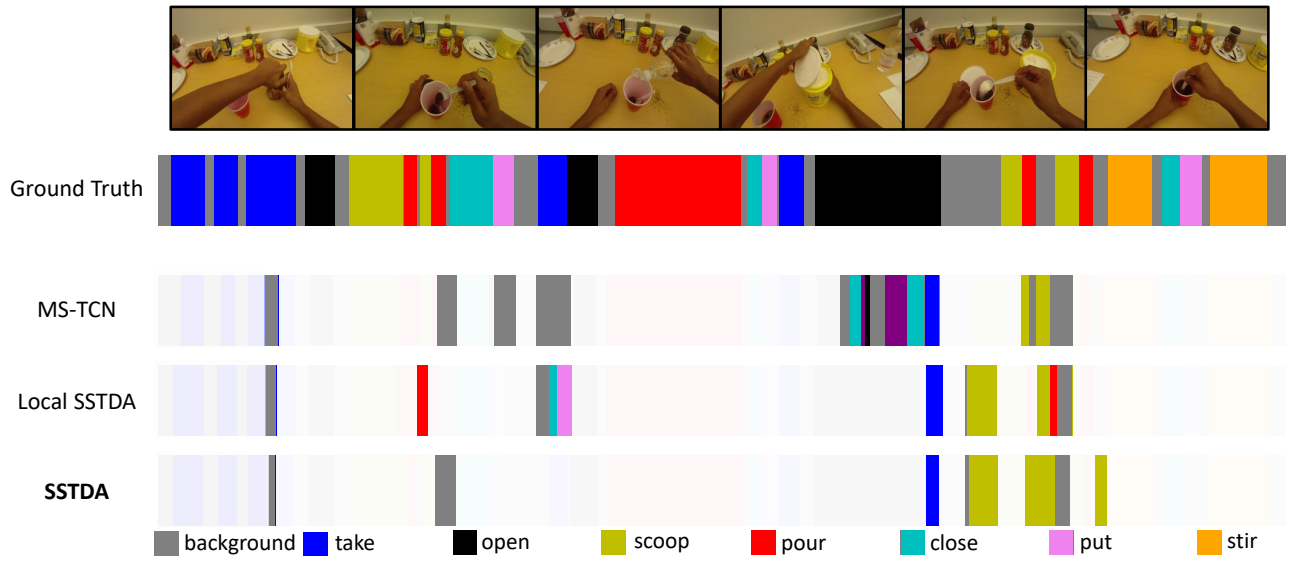
[2] Min-Hung Chen, Zsolt Kira, Ghassan AlRegib, Jaekwon Woo, Ruxin Chen, and Jian Zheng. Temporal attentive alignment for large-scale video domain adaptation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 1

[3] Min-Hung Chen, Baopu Li, Yingze Bao, and Ghassan AlRegib. Action segmentation with mixed temporal domain adaptation. In *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020. 1

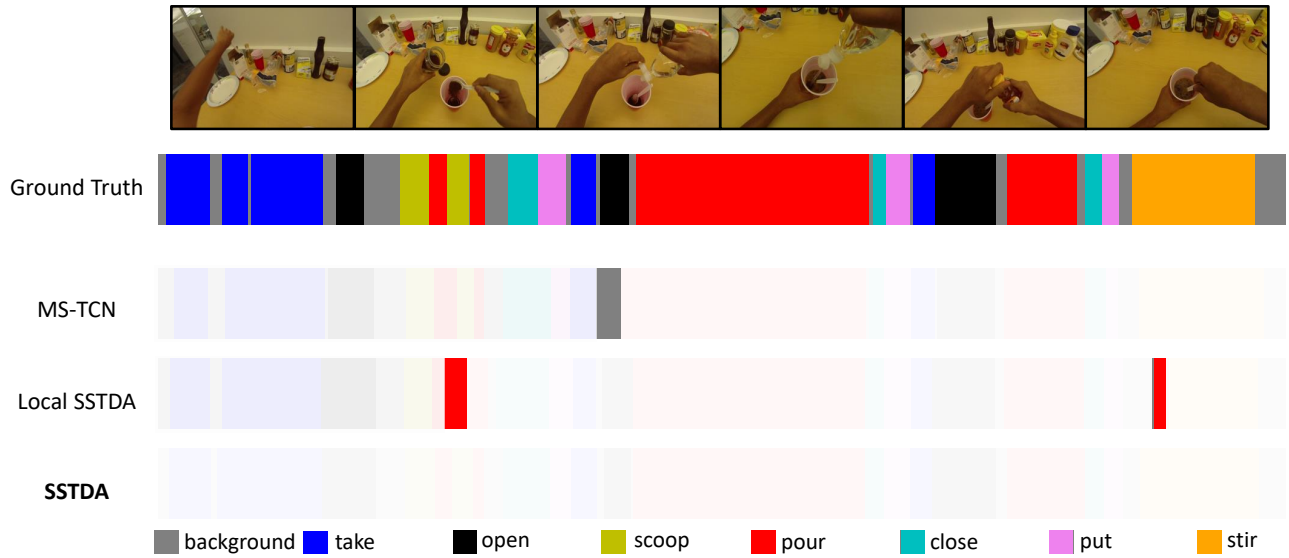
[4] Yazan Abu Farha and Jurgen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 4

[5] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning (ICML)*, 2015. 2

[6] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research (JMLR)*, 17(1):2096–2030, 2016. 2, 3, 4



(a) *Make coffee*



(b) *Make honey coffee*

Figure 3: The visualization of temporal action segmentation for our methods with color-coding on *GTEA*. The video snapshots and the segmentation visualization are in the same temporal order (from left to right). We only highlight the action segments that are significantly different from the ground truth for clear comparison. "MS-TCN" represents the baseline model trained with only source data.

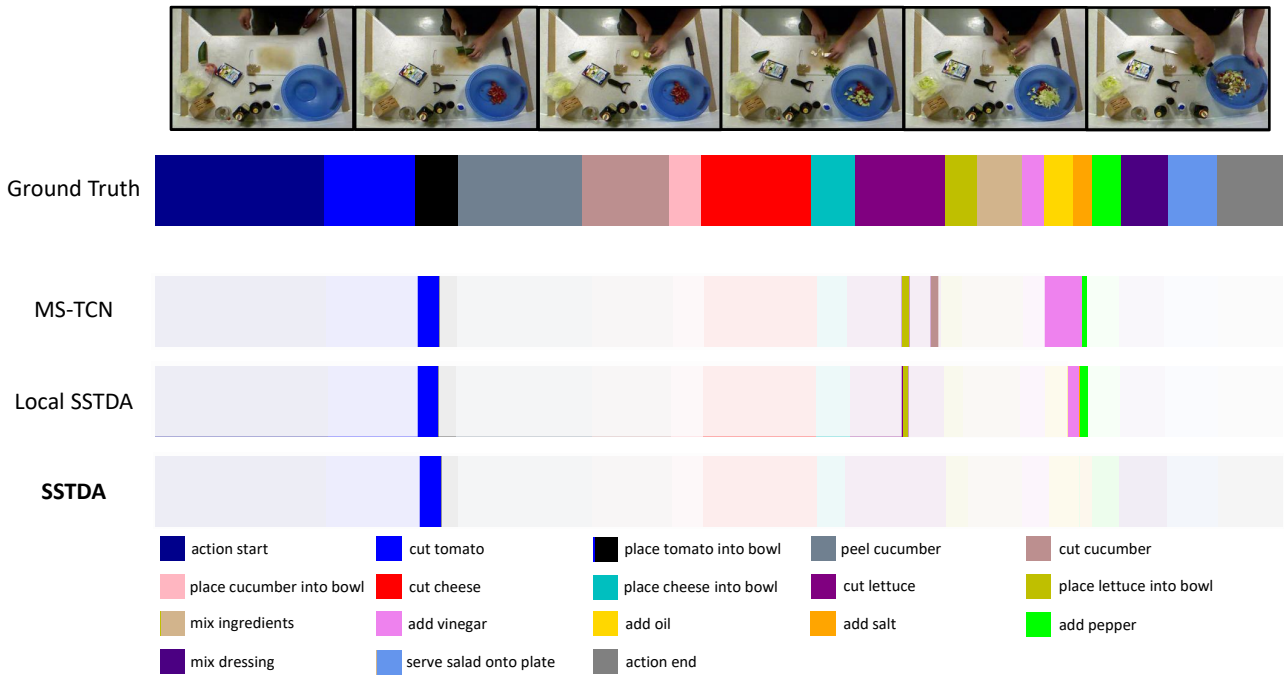
[7] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[8] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 4

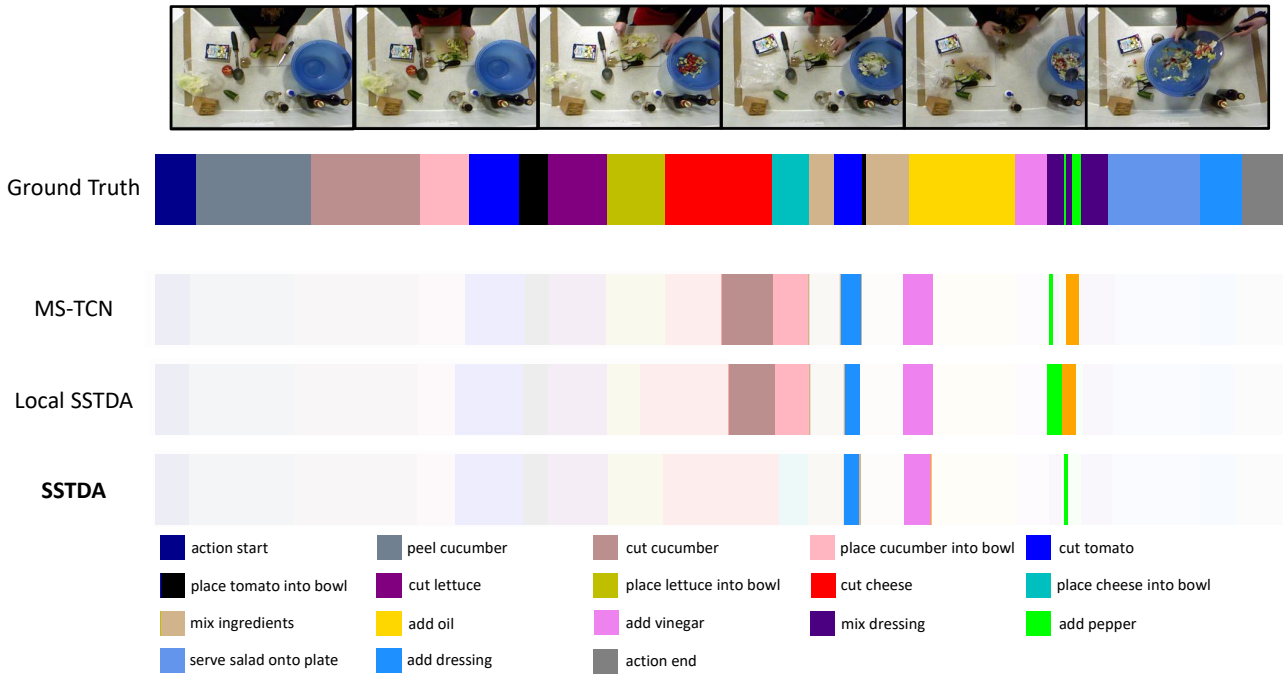
[9] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *International Conference on Machine Learning (ICML)*, 2017. 3, 4

[10] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems Workshop (NeurIPSW)*, 2017. 2

[11] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 3, 4



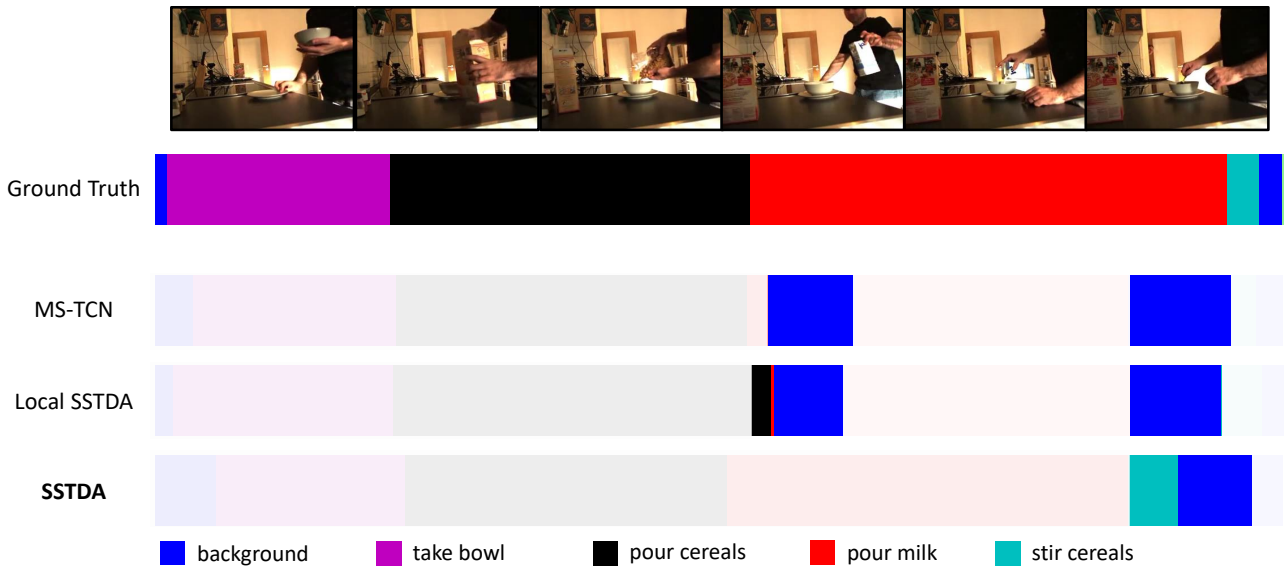
(a) *Subject 02*



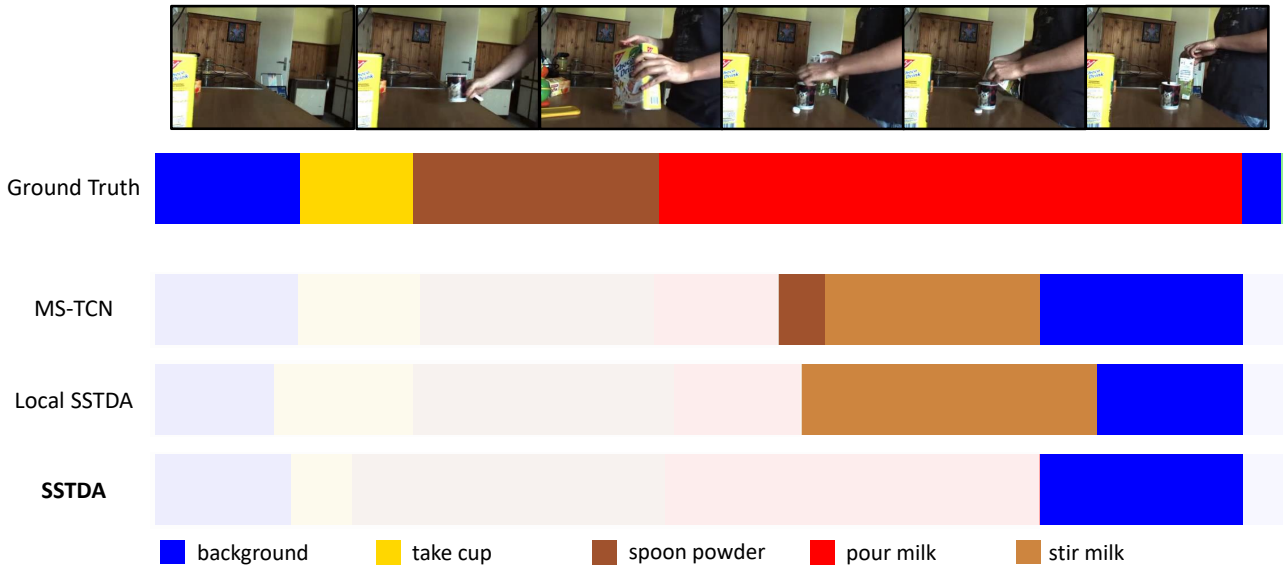
(b) *Subject 04*

Figure 4: The visualization of temporal action segmentation for our methods with color-coding on *50Salads*. The video snapshots and the segmentation visualization are in the same temporal order (from left to right). We only highlight the action segments that are different from the ground truth for clear comparison. Both examples correspond to the same activity *Make salad*, but they are performed by different subjects, i.e., people.

[12] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3, 4



(a) *Make Cereal*



(b) *Make milk*

Figure 5: The visualization of temporal action segmentation for our methods with color-coding on *Breakfast*. The video snapshots and the segmentation visualization are in the same temporal order (from left to right). We only highlight the action segments that are different from the ground truth for clear comparison.

[13] Benoit Steiner, Zachary DeVito, Soumith Chintala, Sam Gross, Adam Paszke, Francisco Massa, Adam Lerer, Gregory Chanan, Zeming Lin, Edward Yang, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2

[14] Ximei Wang, Liang Li, Weirui Ye, Mingsheng Long, and Jianmin Wang. Transferable attention for domain adaptation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019. 1

[15] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In *International Conference on Machine Learning (ICML)*, 2018. 3, 4

[16] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 4