Improving Action Segmentation via Graph Based Temporal Reasoning Supplementary Material

Yifei Huang, Yusuke Sugano, Yoichi Sato
Institute of Industrial Science, The University of Tokyo
{hyf,sugano,ysato}@iis.u-tokyo.ac.jp

In this supplementary material, we first report some additional experiments to examine different variants of our proposed GTRM. Then we show more qualitative results on the 50Salads dataset and the Breakfast dataset. Finally we explain more details of the training process of our model.

1. Additional Experiments

1.1. Comparison with 1D convolution

In our GTRM, we use GCN as a way to perform reasoning on the graph structure. An alternative approach to perform reasoning on the temporal relation of segments is to perform 1D convolution on the sequence of segments. While it can increase the computational cost and potentially aggregate irrelevant information (from, *e.g*., background segments containing no action), 1D convolution can still gather information from the neighbourhood nodes for updating the hidden representation of each node.

EGTEA	**F1**@{10,25,50}			**Edit**	**Acc**
m-GRU	32.6	27.7	17.6	36.0	67.1
C-conv + R-conv	41.5	35.9	24.2	40.7	68.2
C-conv + R-GCN	41.4	35.7	23.2	41.2	68.2
C-GCN + R-conv	**41.7**	36.3	24.2	**42.4**	67.9
C-GCN + R-GCN	41.6	**37.5**	**25.9**	41.8	**69.5**

Table 1. Changing GCN operation to 1D convolution.

In this experiments, we replace either of R-CGN and C-GCN with 1D convolution and compare the performances on the EGTEA dataset. Results are shown in Table 1. **C-conv + R-conv** corresponds to the model where both C-GCN and R-GCN are replaced with 1D convolution. **C-conv + R-GCN** and **C-GCN + R-conv** denote the cases where either the C-GCN or R-GCN is replaced with 1D convolution, respectively. **C-GCN + R-GCN** corresponds to our originally reported result.

Despite the fact that 1D convolution increases the computational complexity especially with larger kernel sizes, the overall performance change remains small. This illustrates the benefit of using GCN rather than 1D convolution on nodes for reasoning the temporal relations of the action segments.

1.2. Comparison with other alternatives tools for modeling relation

Similarly with the previous section, the GCNs in our GTRM could be also changed to other alternatives (*e.g.* LSTM). In this section we discuss the effect on result and computational cost when GCNs are replaced by other alternatives. These include bi-LSTM [2], dilated convolution (dil.-conv) [1] and 1D-convolution (1D-conv) introduced above. We also compare the number of parameters, number of added parameters, and the FLOPs added for each alternative.

variants	**F1**@{10,25,50}			Params	Δ Params	Δ FLOPs
None	32.6	27.7	17.6	1.53M	0	0
bi-LSTM [2]	33.7	28.5	17.9	2.38M	0.85M	23%
dil.-conv [1]	39.8	34.0	22.6	2.50M	0.97M	20%
1D-conv	41.5	35.9	24.2	3.79M	2.26M	47%
GTRM	**41.6**	**37.5**	**25.9**	1.85M	0.32M	8%

Table 2. Number (and added number) of parameters, added FLOPs of different variants of GTRM on the EGTEA dataset.

Table 2 shows the number of parameters and its increase from the baseline (None), together with the increase in FLOPs. Overall, GCN not only achieves the best performance but is also significantly faster and requires much fewer parameters than other methods such as 1D convolution and recurrent network. As shown in the table, our GTRM uses much fewer parameters and is significantly faster than the best baseline (1D-conv). All the other baselines mentioned in the previous paragraph also take up much more ($\sim 3\times$ more) computational time than our proposed model (except FCN).

1.3. Influence of edge weighting

In this section we discuss the influence of different edge weight designs in R-GCN and C-GCN. While in our GTRM edges are weighted according to the temporal distances, it is also possible to use uniform weights on each edge which still enables message passing between connected nodes.

We replace the edge weight of either C-GCN or R-GCN to uniform weighting and compare the performance on the EGTEA dataset. Experimental results are summarized in Table 3. **C-Uni+R-Uni** is the case where both GCNs use uniform weight. **C-Uni+R-GCN** and **C-GCN+R-Uni** indicates the cases where only edges in C-GCN or R-GCN is changed to uniform weight, respectively. **C-GCN+R-GCN** is our reported result.

Gain	**F1**@{10,25,50}			**Edit**	**Acc**
m-GRU	32.6	27.7	17.6	36.0	67.1
C-Uni + R-Uni	39.6	34.2	23.2	41.3	67.4
C-Uni + R-GCN	40.9	35.9	24.8	41.4	67.4
C-GCN + R-Uni	40.6	35.8	24.1	41.2	67.4
C-GCN + R-GCN	**41.6**	**37.5**	**25.9**	**41.8**	**69.5**

Table 3. Changing edge weight to uniform weight.

We can see that, even with the uniform edge weights, the performance is far better than the baseline m-GRU. This demonstrates the importance of modeling temporal relations among action segments for a better action segmentation. The performance is further improved with our proposed edge weighting scheme, and supports the effectiveness of the distance-based weight design.

2. Qualitative results

In this section, we show more qualitative results on the Breakfast dataset (Fig 1) and the 50Salads dataset (Fig 2). In both figures, we show the ground truth segmentation results in the first row (a), results of MS-TCN in the second row (b), and the result of MS-TCN with our proposed GTRM on top in the third row (c). We can see that in these two datasets, while adding our proposed GTRM on top of MS-TCN improve the overall result, the influence is relatively small.

3. Training Details

For training the model, we use the Adam optimizer with its default settings. We summarize the training of our model in 3 stages. In the first stage, we first train the backbone model for 50 epochs to get a reasonable result for the proposed GTRM. During this stage, the loss only includes $\mathcal{L}_{\text{seg}}(\boldsymbol{y}_i^{gt}, \boldsymbol{y}_i)$. Then in the second stage, we then fix the backbone and train the GTRM for 50 epochs. During this stage, we alternatively feed the ground truth action segments Y^{gt} and the output from the backbone Y as input to our GTRM. The loss function is the joint loss function of Equation 8 in the submission without the first item $\mathcal{L}_{\text{seg}}(\boldsymbol{y}_i^{gt}, \boldsymbol{y}_i)$. The learning rate in the first two stages are both 5×10^{-4}. In the final stage, we jointly train the whole model under the joint loss function, with a reduced learning rate of 1×10^{-4} In all experiments, we set $\lambda_t = 0.15$, $\lambda_1 = \lambda_2 = 0.5$ for loss functions.

References

[1] Yazan Abu Farha and Jurgen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[2] Bharat Singh, Tim K Marks, Michael Jones, Oncel Tuzel, and Ming Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

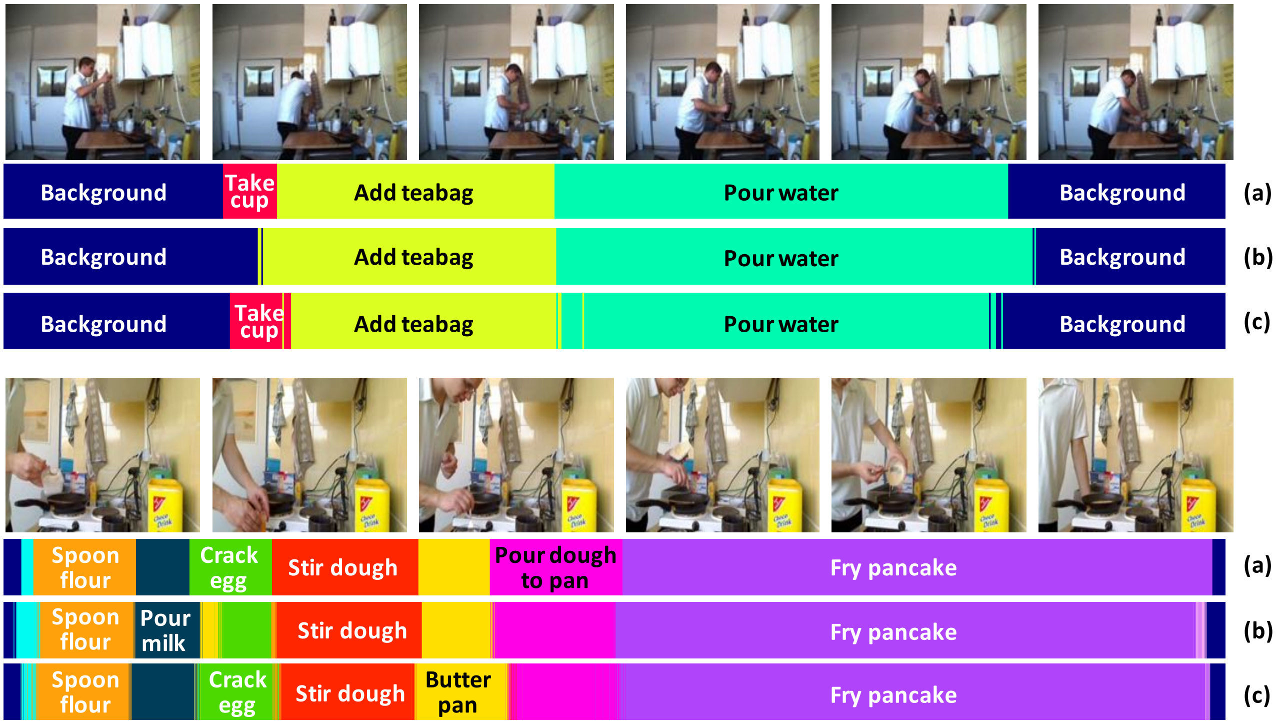

Figure 1. Qualitative results on the Breakfast dataset. Result is shown in full video length. Corresponding results are: (a) Ground truth, (b) MS-TCN, (c) MS-TCN + GTRM.

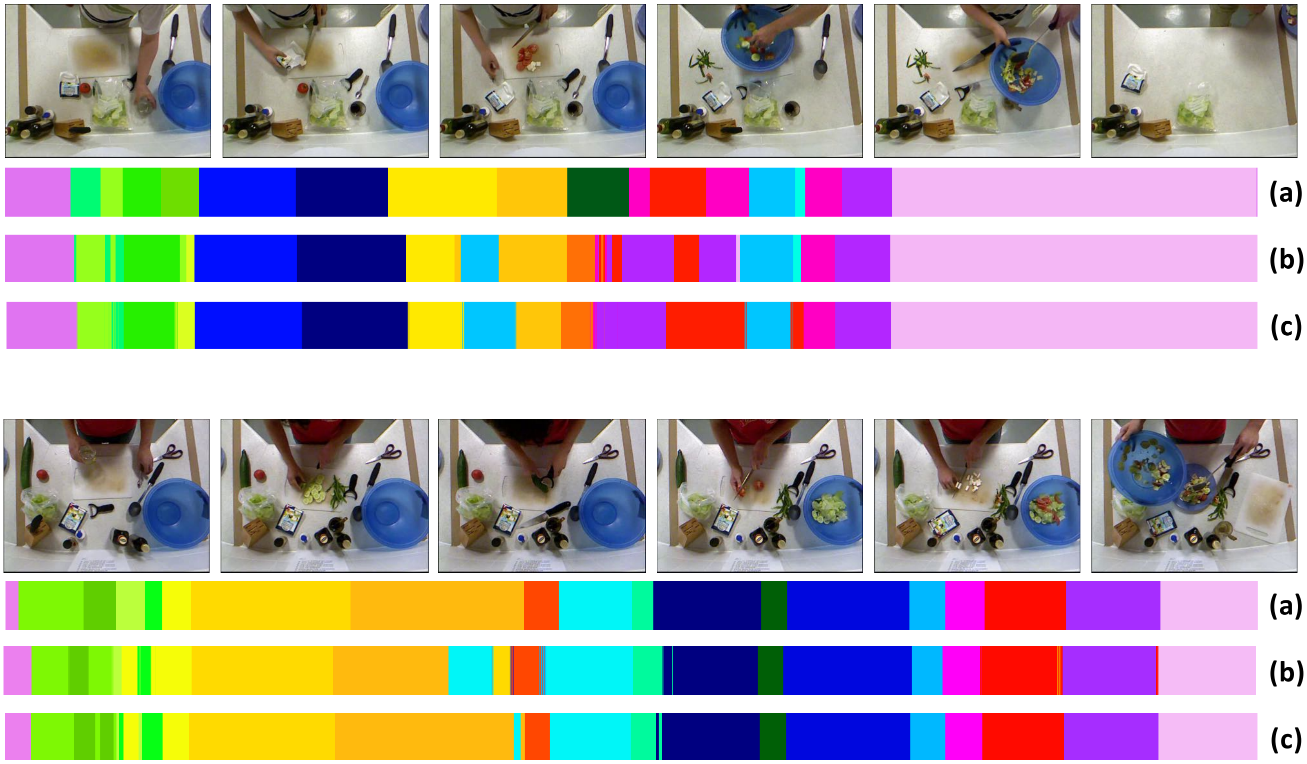

Figure 2. Qualitative results on the 50Salads dataset. Result is shown in full video length. Corresponding results are: (a) Ground truth, (b) MS-TCN, (c) MS-TCN + GTRM.