Supplementary Material for Learning Motion in Feature Space: Locally-Consistent Deformable Convolution Networks for Fine-Grained Action Detection

A. Full formulation for convolutions with multiple output channels

Suppose that the input $\mathbf{x}$ of a convolution has I channels and the output has O channels, *i.e.* $\mathbf{x} \in R^{N\times I}, \mathbf{y} \in \mathbb{R}^{M\times O}$, we can write the standard convolution as:

$$\mathbf{y}_j[n] = \sum_i \sum_k \mathbf{w}_{j,i}[-k]\mathbf{x}_i\left[n+k\right], \tag{11}$$

where $i \in \{1,\dots,I\}, j \in \{1,\dots,O\}$, and $\mathbf{w} \in \mathbb{R}^{N\times K\times I\times O}$. The original deformable convolution, therefore, is written as:

$$\mathbf{y}_j[n] = \sum_i \sum_k \mathbf{w}_{j,i}[-k]\mathbf{x}_i\left(n+k+\ddot{\Delta}_{n,k}\right), \tag{12}$$

In reality, there are multiple deformable groups ($G > 1$), meaning that different input channels can have different deformation offsets. Specifically, a multi-channel deformable convolution with multiple deformable group can be written as:

$$\mathbf{y}_j[n] = \sum_i \sum_k \mathbf{w}_{j,i}[-k]\mathbf{x}_i\left(n+k+\ddot{\Delta}_{g_i,n,k}\right), \tag{13}$$

where g_i is the deformable group that the input channel i belongs to. We keep the deformable group as $G = 1$ and drop the notation g_i for the sake of simplicity.

We write the multi-channel LCDC as:

$$\mathbf{y}_j[n] = \sum_i \sum_k \mathbf{w}_{j,i}[-k]\mathbf{x}_i\left(n+k+\dot{\Delta}_{n+k}\right). \tag{14}$$

It is equivalent to

$$\mathbf{y}_j[n] = \sum_i \sum_k \mathbf{w}_{j,i}[-k]\tilde{\mathbf{x}}_i[n+k] = (\tilde{\mathbf{x}} * \mathbf{w}_j)[n], \tag{15}$$

where

$$\tilde{\mathbf{x}}_i[n] = (D_{\dot{\Delta}}\{\mathbf{x}_i\})[n] = \mathbf{x}_i\left(n+\dot{\Delta}_n\right). \tag{16}$$

B. More reasoning on the difference of receptive fields

$\ddot{\mathbf{r}}^{(t)}$ and $\dot{\mathbf{r}}^{(t)}$ of deformable convolution and locally-consistent deformable convolution *carries temporal information* because the offsets are constructed from inputs at different time frames. This property is not valid in other types of convolutions. We can write standard convolutions and dilated convolutions as special cases of deformable convolutions, *i.e.* $\ddot{\Delta} = 0$ in standard convolution and $\ddot{\Delta}^{(t)} = const, \forall t$. Hence,

- Standard convolution:
$$\ddot{\Delta}^{(t)} = 0, \forall t \Rightarrow \ddot{\mathbf{r}}^{(t)} = 0, \forall t,$$
- Dilated convolution:
$$\ddot{\Delta}^{(t)} = \ddot{\Delta}^{(t-1)}, \forall t \Rightarrow \ddot{\mathbf{r}}^{(t)} = 0, \forall t.$$
- Deformable convolution:
$$\ddot{\Delta}^{(t)} \neq \ddot{\Delta}^{(t-1)} \Rightarrow \ddot{\mathbf{r}}^{(t)} \neq 0.$$
- Locally-consistent deformable convolution:
$$\dot{\Delta}^{(t)} \neq \dot{\Delta}^{(t-1)} \Rightarrow \dot{\mathbf{r}}^{(t)} \neq 0.$$

C. In-detail architecture of LCDC

Tab. 4 shows the detailed architecture implementation of LCDC.

Layer	Input(s)	Output size	Kernel size	Comments
conv1	data	(112,112,64)	(7,7,64), stride2	with 3,3 maxpool, stride2 frames of all snippets are unrolled
conv2x	bn_conv1	(56,56,256)	$\begin{bmatrix} 1,1,64 \\ 3,3,64 \\ 1,1,256 \end{bmatrix} \times 3$	input is output of conv1
conv3x	res2c_relu	(28,28,512)	$\begin{bmatrix} 1,1,128 \\ 3,3,128 \\ 1,1,512 \end{bmatrix} \times 4$	input is output of conv2x
conv4x	res3d_relu	(14,14,1024)	$\begin{bmatrix} 1,1,256 \\ 3,3,256 \\ 1,1,1024 \end{bmatrix} \times 6$	input is output of conv3x
res5a_branch1	res4f_relu	(14,14,2048)	(1,1,2048)	input is output of conv4x
bn5a_branch1	(prev)	(14,14,2048)	-	batch normalization
res5a_branch2a	(prev)	(14,14,512)	(1,1,512)	convolution
bn5a_branch2a	(prev)	(14,14,512)	-	batch normalization
res5a_branch2a_relu	(prev)	(14,14,512)	-	ReLU
res5a_branch2b_offset	(prev)	(14,14,2)	(3,3,2)	offset learner
res5a_branch2b_offset_expand	(prev)	(14,14,18)	-	expand by replication
res5a_branch2b	res5a_branch2a_relu res5a_branch2b_offset_expand	(14,14,512)	(3,3,512)	deformable convolution
bn5a_branch2b	(prev)	(14,14,512)	-	batch normalization
res5a_branch2b_relu	(prev)	(14,14,512)	-	ReLU
res5a_branch2c	(prev)	(14,14,2048)	(1,1,2048)	convolution
bn5a_branch2c	(prev)	(14,14,2048)	-	batch normalization
res5a	bn5a_branch1 bn5a_branch2c	(14,14,2048)	-	addition
res5a_relu	(prev)	(14,14,2048)	-	ReLU
res5b_branch2a	(prev)	(14,14,512)	(1,1,512)	convolution
bn5b_branch2a	(prev)	(14,14,512)	-	batch normalization
res5b_branch2a_relu	(prev)	(14,14,512)	-	ReLU
res5b_branch2b_offset	(prev)	(14,14,2)	(3,3,2)	offset learner
res5b_branch2b_offset_expand	(prev)	(14,14,18)	-	expand by replication
res5b_branch2b	res5b_branch2a_relu res5b_branch2b_offset_expand	(14,14,512)	(3,3,512)	deformable convolution
bn5b_branch2b	(prev)	(14,14,512)	-	batch normalization
res5b_branch2b_relu	(prev)	(14,14,512)	-	ReLU
res5b_branch2c	(prev)	(14,14,2048)	(1,1,2048)	convolution
bn5b_branch2c	(prev)	(14,14,2048)	-	batch normalization
res5b	res5a_relu bn5b_branch2c	(14,14,2048)	-	addition
res5b_relu	(prev)	(14,14,2048)	-	ReLU
res5c_branch2a	(prev)	(14,14,512)	(1,1,512)	convolution
bn5c_branch2a	(prev)	(14,14,512)	-	batch normalization
res5c_branch2a_relu	(prev)	(14,14,512)	-	ReLU
res5c_branch2b_offset	(prev)	(14,14,2)	(3,3,2)	offset learner
res5c_branch2b_offset_expand	(prev)	(14,14,18)	-	expand by replication
res5c_branch2b	res5c_branch2a_relu res5c_branch2b_offset_expand	(14,14,512)	(3,3,512)	deformable convolution
bn5c_branch2b	(prev)	(14,14,512)	-	batch normalization
res5c_branch2b_relu	(prev)	(14,14,512)	-	ReLU
res5c_branch2c	(prev)	(14,14,2048)	(1,1,2048)	convolution
bn5c_branch2c	(prev)	(14,14,2048)	-	batch normalization
res5c	res5b_relu bn5c_branch2c	(14,14,2048)	-	addition
res5c_relu	(prev)	(14,14,2048)	-	ReLU
conv_new_1	(prev)	(14,14,256)	(1,1,256)	convolution
conv_new_1_relu	(prev)	(14,14,256)	-	ReLU
spacetime_fusion	conv_new_1_relu res5a_branch2b_offset res5b_branch2b_offset res5c_branch2b_offset	(L-1,14,14,262)	-	reshape all frames back into snippets, then concatenate difference of all offset layers with conv_new_1_relu
spacetime_conv1	(prev)	(L-1,14,14,256)	(4,3,3,256)	3Dconv with window size for temporal dimension of 4
spacetime_bn1	(prev)	(L-1,14,14,256)	-	batch normalization
spacetime_relu1	(prev)		-	ReLU

spacetime_pool1	(prev)	((L-1)/2,14,14,256)	-	temporal max pooling of size 2
spacetime_conv2	(prev)	((L-1)/2,14,14,256)	(4,3,3,256)	3Dconv with window size for temporal dimension of 4
spacetime_bn2	(prev)	((L-1)/2,14,14,256)	-	batch normalization
spacetime_relu2	(prev)		-	ReLU
spacetime_pool2	(prev)	((L-1)/4,14,14,256)	-	temporal max pooling of size 2
spacetime_reduce	(prev)	(14,14,256)	-	averaging across time domain
pool_new	(prev)	(7,7,256)	-	max pooling, stride 2
fc_new_1	(prev)	(1024)	-	fully connected with ReLU
fc_new_2	(prev)	(1024)	-	fully connected with ReLU

Table 4: LCDC architecture in detail. The groups conv1, conv2x, conv3x, and conv4x are the same as the original ResNet50. The convention of kernel size: (kernel_height, kernel_width, number_of_output_channels) for 2D convolution and (kernel_time, kernel_height, kernel_width, number_of_output_channels) for 3D convolution. Size of input data is (224, 224, 3). L is the number of frames per video snippet (we choose $L = 16$). If the input is annotated as *(prev)*, it means it uses the output from the previous layer.

D. In-detail figures

We provide higher-resolution versions of Fig. 2, Fig. 4, and Fig. 3 in Fig. 6, Fig. 7, and Fig. 8 respectively. Fig. 9 and Fig. 10 also show higher-resolution versions of Fig. 5 with annotation of color-code. We also provide the groundtruth action sequence of the two videos. Readers can view the videos corresponding to Fig. 9 and Fig. 10 in other additional supplementary materials (*50salads.mp4* and *gtea.mp4*).

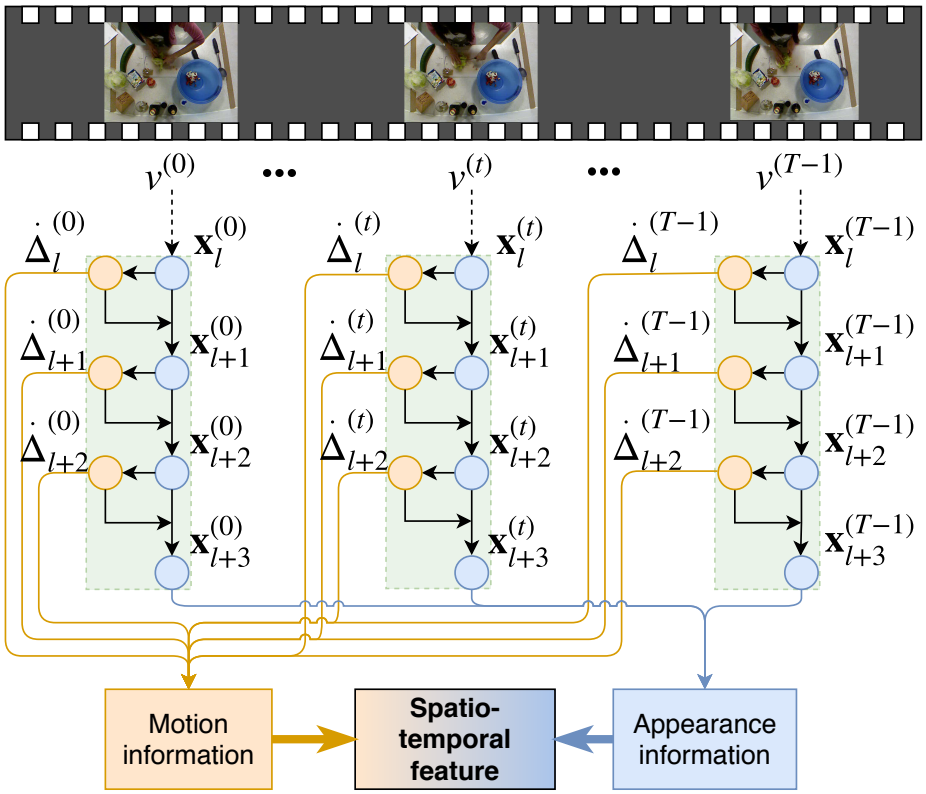

Figure 6: Network architecture of our proposed framework across multiple frames $v^{(t)}$. Appearance information comes from the last layer while motion information is extracted directly from deformation $\dot{\Delta}$ in the feature space instead of from a separate optical flow stream. Weights are shared across frames over time.

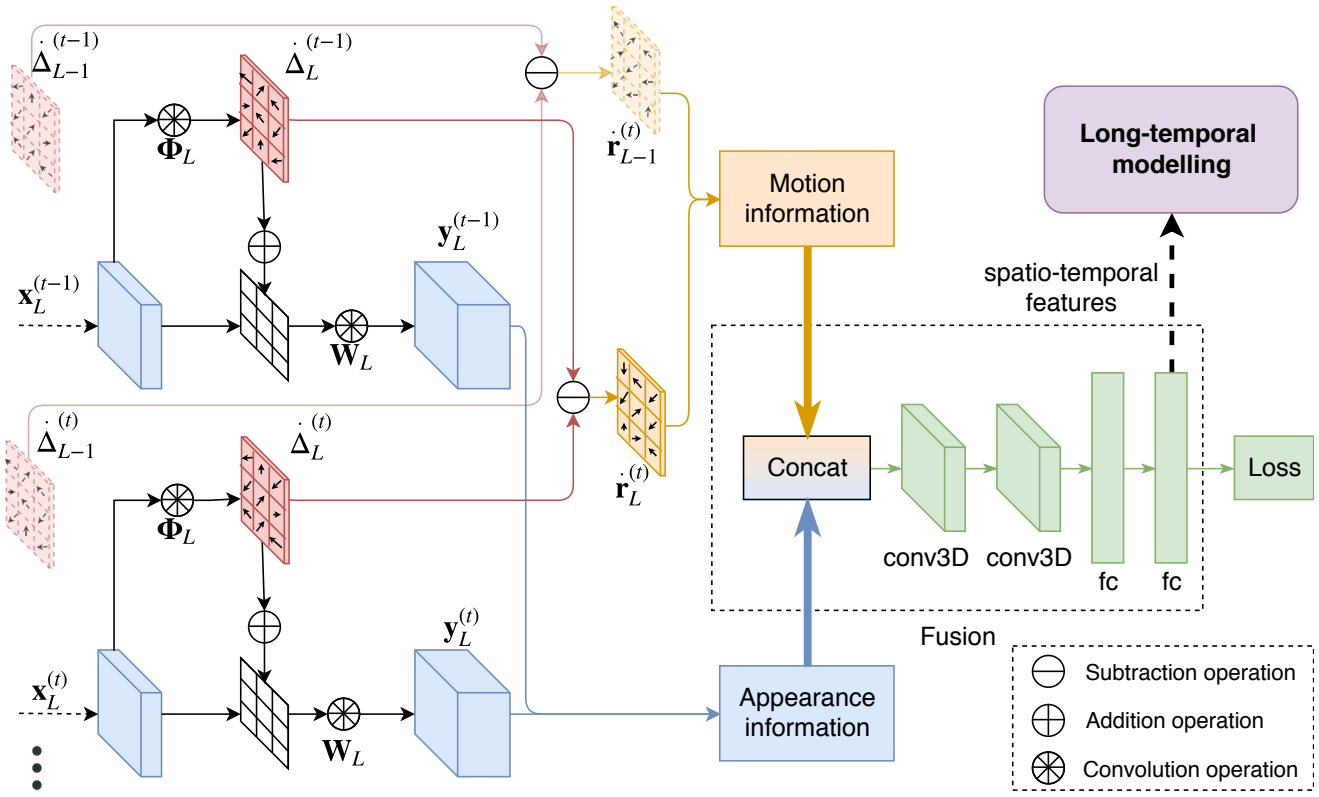

Figure 7: A more detailed view of our network architecture with the fusion module. Appearance information comes from output of the last layer while motion information comes from aggregating $\dot{\mathbf{r}}$ from multiple layers. Outputs of the final fc layer can be flexibly used as the features for any long-temporal modeling networks.

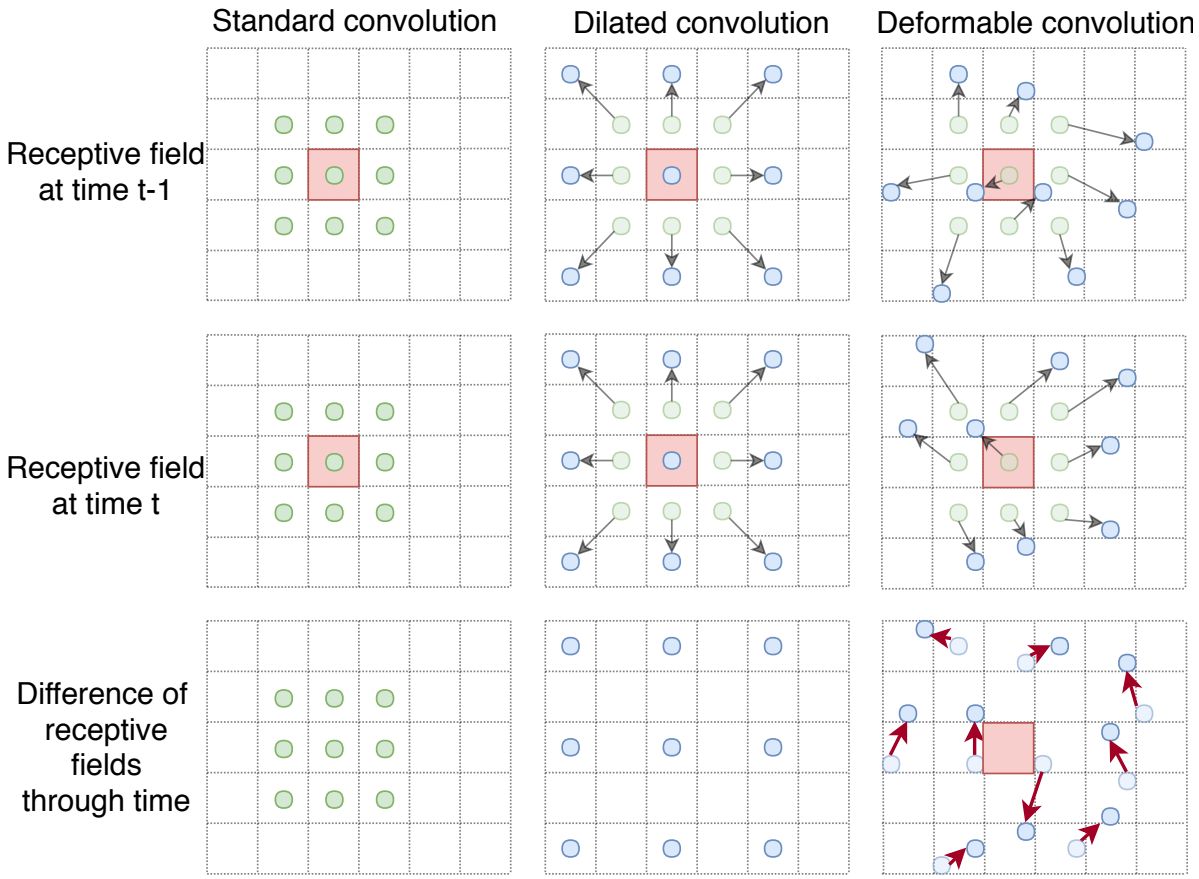

Figure 8: Illustration of temporal information modeled by the difference of receptive fields at a single location in 2D. Only deformable convolution can capture temporal information (shown with red arrows). Related to Eq. (2) and Eq. (3), n is red square, $n+k$ are green dots, $\ddot{\Delta}_{n,k}$ are black arrows, $n+k+\ddot{\Delta}_{n,k}$ are blue dots, and $\ddot{\mathbf{r}}$ are red arrows.

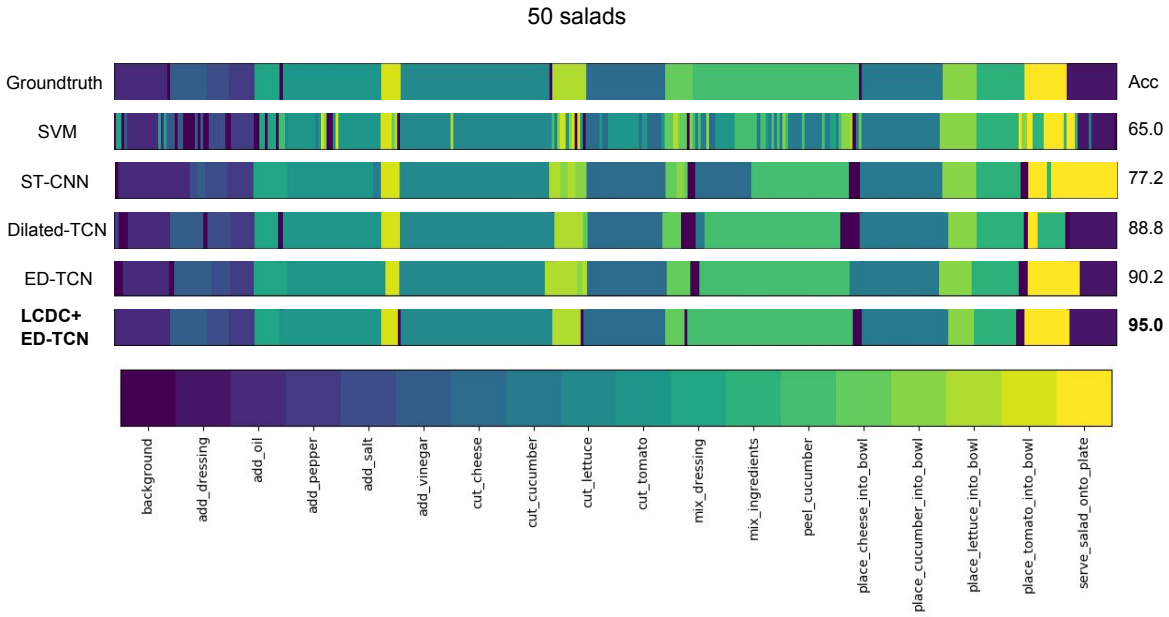

Figure 9: Comparison of segmentation results across different methods on a test video from 50 Salads dataset (*mid*-level). The action sequence is: *add_oil, background, add_vinegar, add_salt, add_pepper, mix_dressing, background, cut_tomato, place_tomato_into_bowl, cut_lettuce, background, place_lettuce_into_bowl, cut_cheese, place_cheese_into_bowl, peel_cucumber, background, cut_cucumber, place_cucumber_into_bowl, mix_ingredients, serve_salad_onto_plate, add_dressing*.

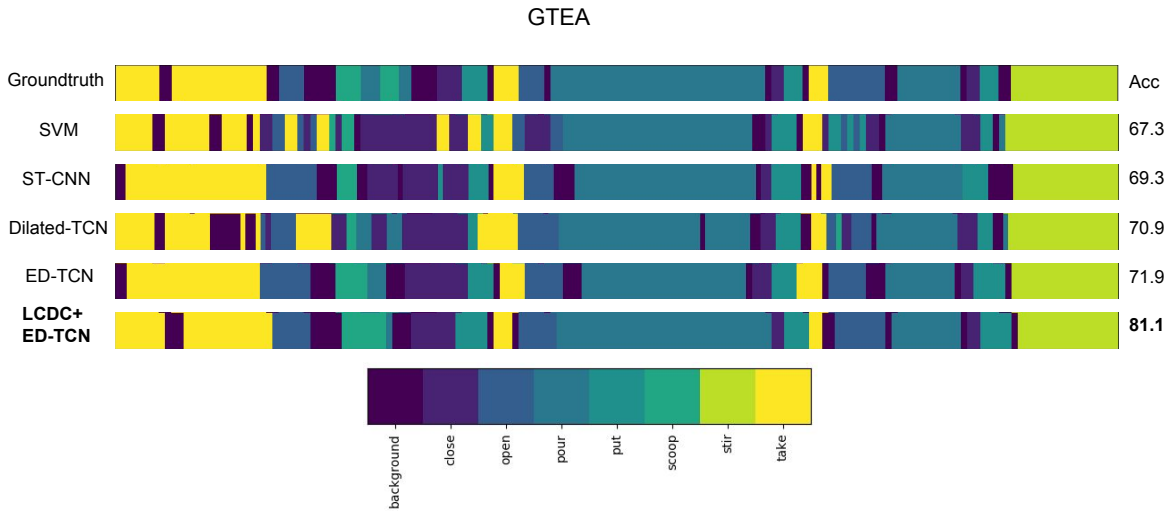

Figure 10: Comparison of segmentation results across different methods on a test video from GTEA dataset. The action sequence is: *take, background, take, background, open, background, scoop, pour, scoop, pour, background, close, put, background, take, open, background, pour, background, close, put, background, take, open, background, pour, background, close, put, background, stir*.