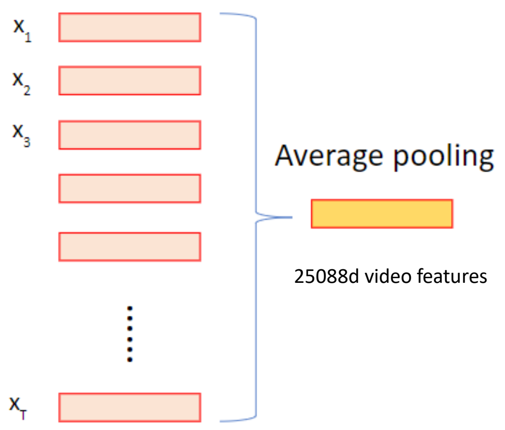
Deep Learning for Computer Vision Hw4

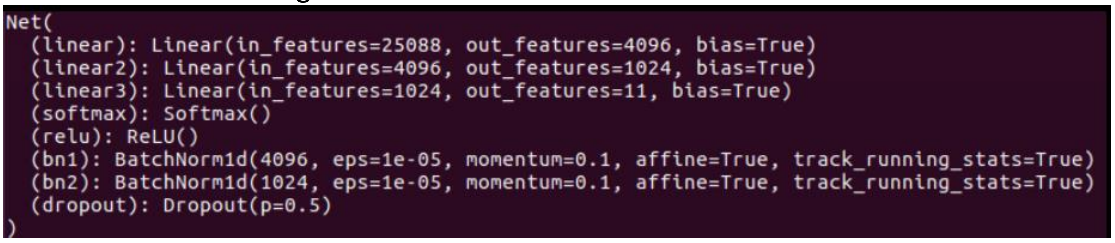
電機所 碩一 r07921010 余奇安

1. Data Preprocessing and CNN features for action recognition

I use vgg16 to extract CNN-based video features which is in 512*7*7 dimension. The input of vgg16 are video frames with down sample factor 12 and all the image normalize to mean 0.485, 0.456, 0.406, std 0.229, 0.224, 0.225 and resize to 224*224 in RGB. After pass the vgg16, the output dimension is (number of frames, 512,7,7), I do the mean pooling on the first dimension and get (512*7*7) to represent each video in 25088 dimensions.

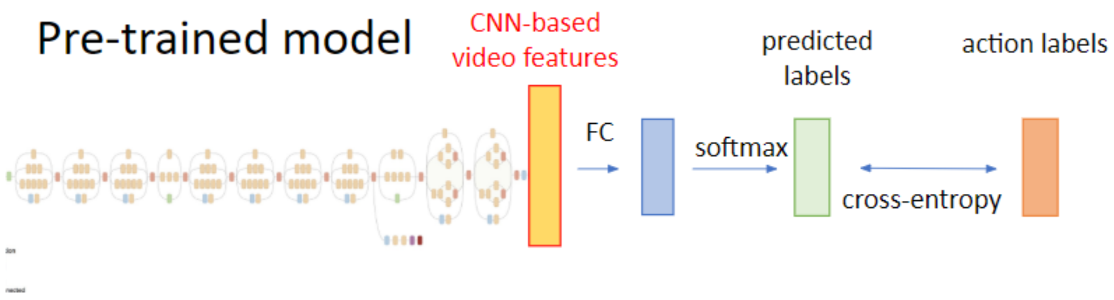


After pooling the CNN-based video features are input to classifier, the classifier structure is as following:

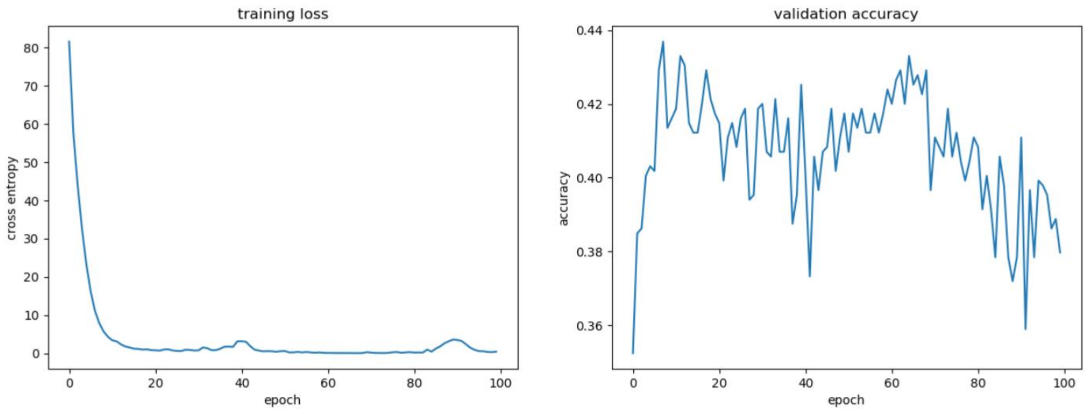


```
Net(
  (linear): Linear(in_features=25088, out_features=4096, bias=True)
  (linear2): Linear(in_features=4096, out_features=1024, bias=True)
  (linear3): Linear(in_features=1024, out_features=11, bias=True)
  (softmax): Softmax()
  (relu): ReLU()
  (bn1): BatchNorm1d(4096, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (bn2): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (dropout): Dropout(p=0.5)
)
```

The input pass, batchnorm, relu, dropout and softmax and calculate cross entropy loss with ground truth.

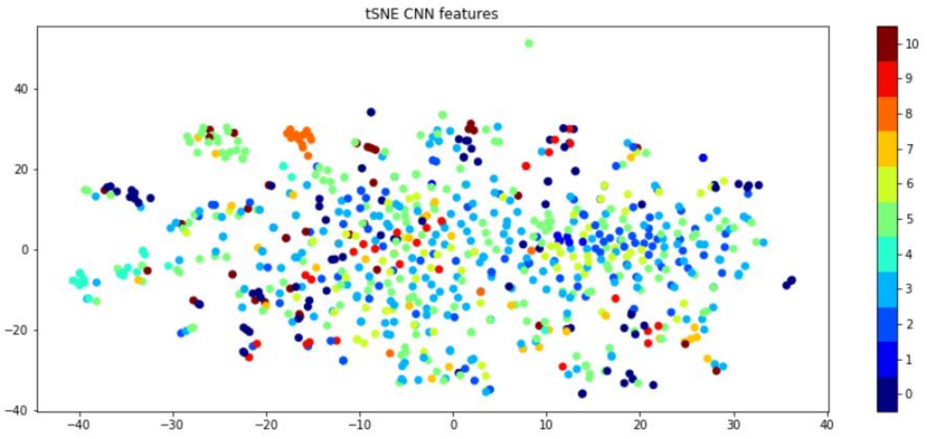


The training curve:



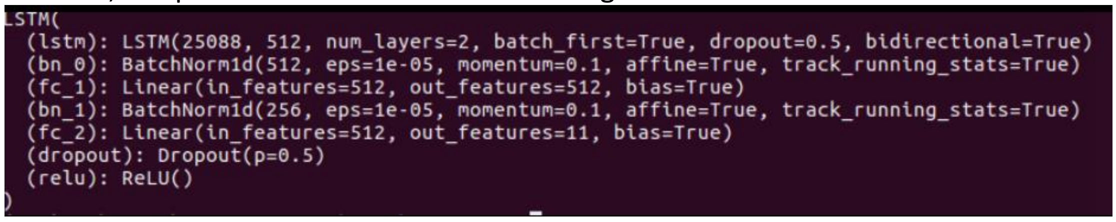
Result:
The best accuracy is 0.436931 on validation set.

Visualization:
t-SNE plot of validation set after vgg16 feature extractor



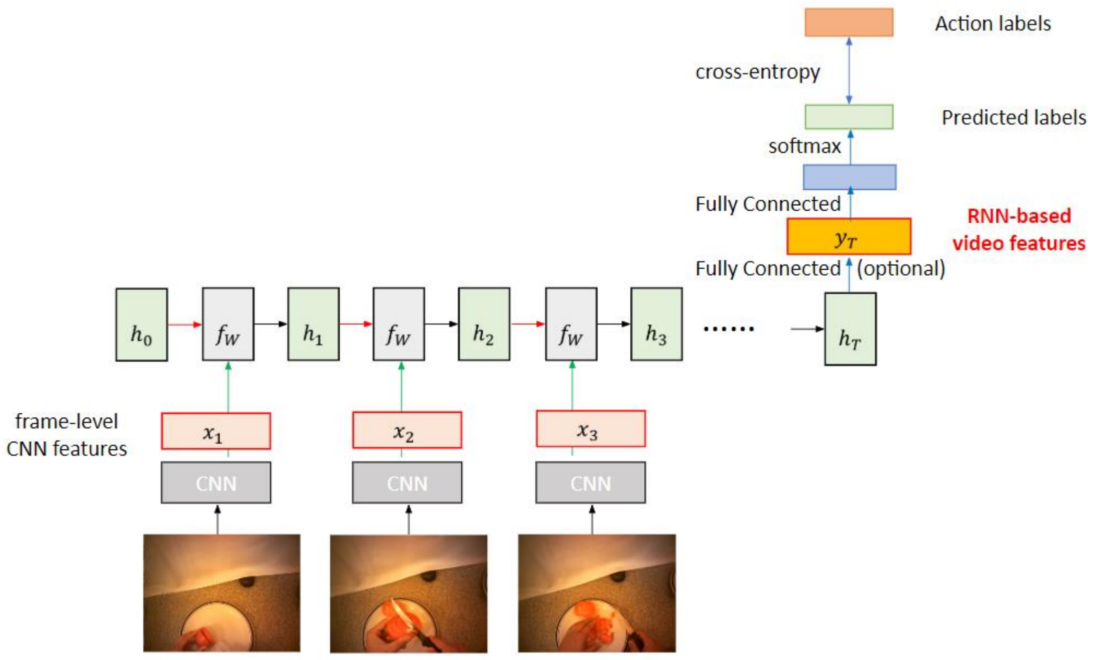
2. Trimmed action recognition (RNN model)

For RNN model, I use the same pretrained model(vgg16) to extract CNN-based features, but don't execute max pooling. I use 2 layers forward LSTM as rnn encoder, the parameter details are as following:

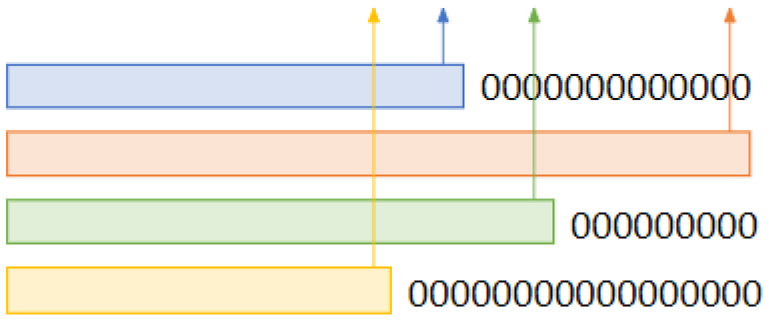


```
LSTM(
  (lstm): LSTM(25088, 512, num_layers=2, batch_first=True, dropout=0.5, bidirectional=True)
  (bn_0): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc_1): Linear(in_features=512, out_features=512, bias=True)
  (bn_1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc_2): Linear(in_features=512, out_features=11, bias=True)
  (dropout): Dropout(p=0.5)
  (relu): ReLU()
)
```

The model architecture referenced from slides:

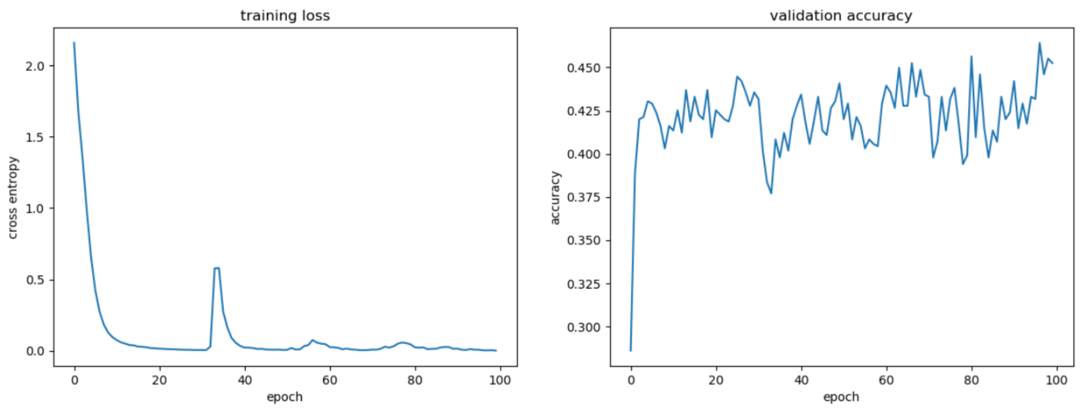


The input dimension of LSTM is (batch size, frames length, 25088); I use pack_padded_sequence to get the last output of each sequences in batch.



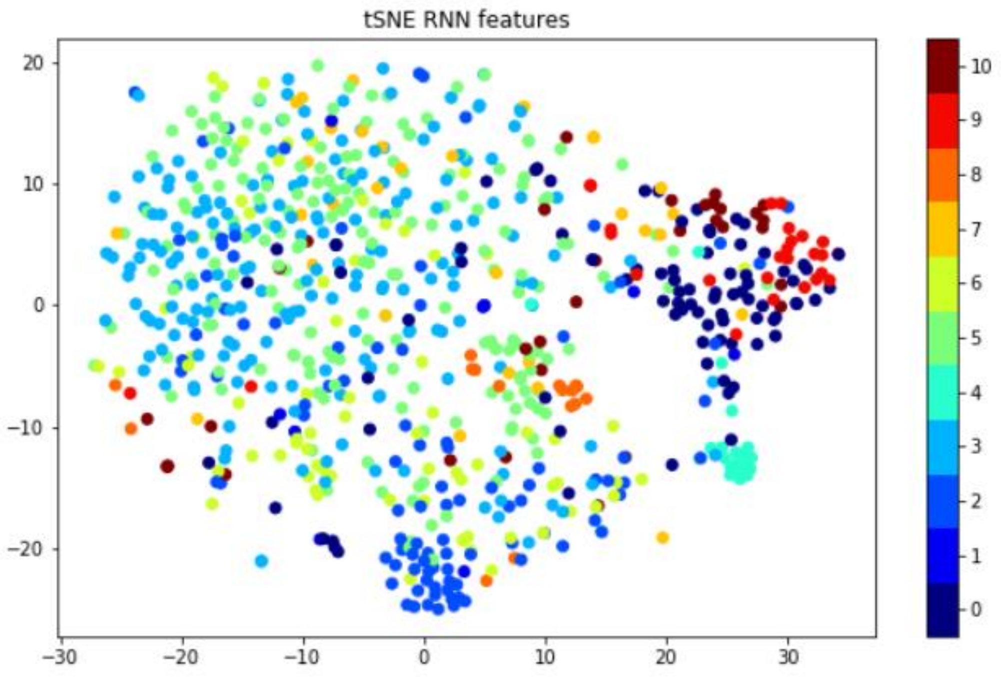
The LSTM encoded feature to classifier is (batch size, 512). After the classifier, the output is (batch size, 11). Finally, calculate the cross entropy between the predict label and ground truth.

Training curve:

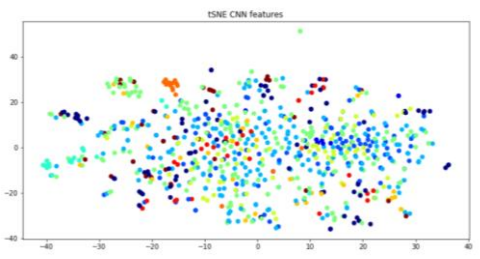
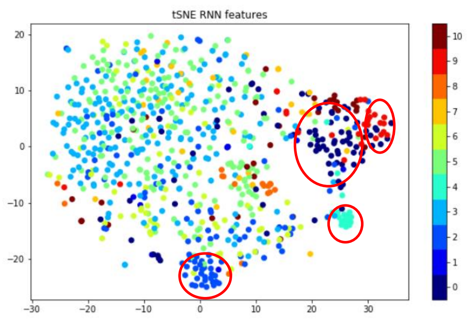


Result:
The best accuracy of validation set is 0.469

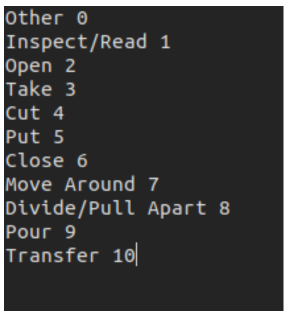
Visualization of RNN encode feature:



Comparison between CNN and RNN feature:

CNN	RNN
	

We can see that RNN feature in class 1, 2, 4, 9 is much closer in cluster.
The reason why RNN feature can improve the performance is that it considers the sequence information. For example, before someone open/cut the food, people usually read/inspect the food at first. The CNN feature only consider the image information while the LSTM model can learn this kinds of sequence information so that the accuracy can increase and the cluster can be more clear on these classes.



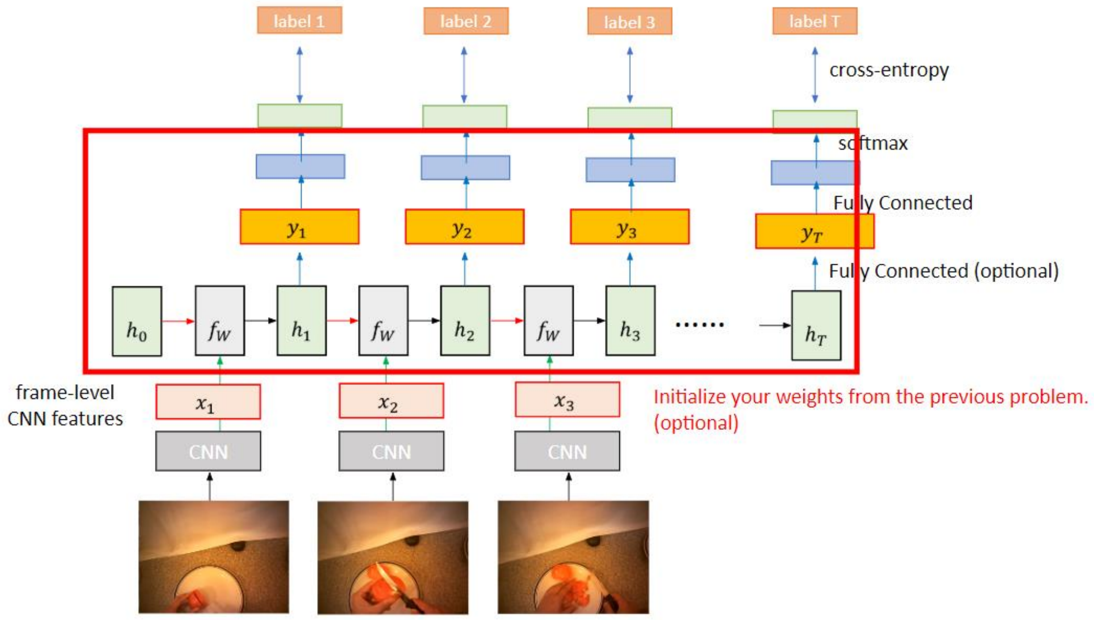
```
Other 0
Inspect/Read 1
Open 2
Take 3
Cut 4
Put 5
Close 6
Move Around 7
Divide/Pull Apart 8
Pour 9
Transfer 10
```

3. Temporal action segmentation (sequence to sequence model)

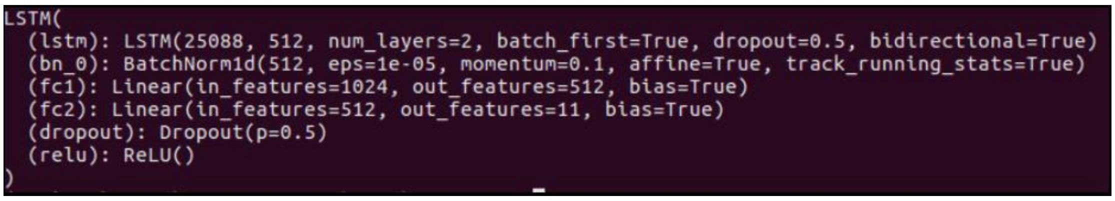
For sequence to sequence model, I sample 512 frames in order for each video. In training process, the input of LSTM encoder is in dimension (batch size, 512, 25088), the output dimension of LSTM is (batch size, 512, 512). Using this sequence feature to classifier and the output of classifier is (batch size, 512, 11). The advantage of sample is that I don't need to deal with padding problems and the sample method also prevent model from overfitting.

For prediction in validation set, I set the batch size equal to 1, which means load one video. The input of LSTM model is (1, frames of video, 25088) and the output is (1, frames of video, 512). Eventually, passing the classifier to predict (frames of video, 11) label.

Model structure reference by slides:



The LSTM encoder structure:



```
LSTM(
  (lstm): LSTM(25088, 512, num_layers=2, batch_first=True, dropout=0.5, bidirectional=True)
  (bn_0): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc1): Linear(in_features=1024, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=11, bias=True)
  (dropout): Dropout(p=0.5)
  (relu): ReLU()
)
```

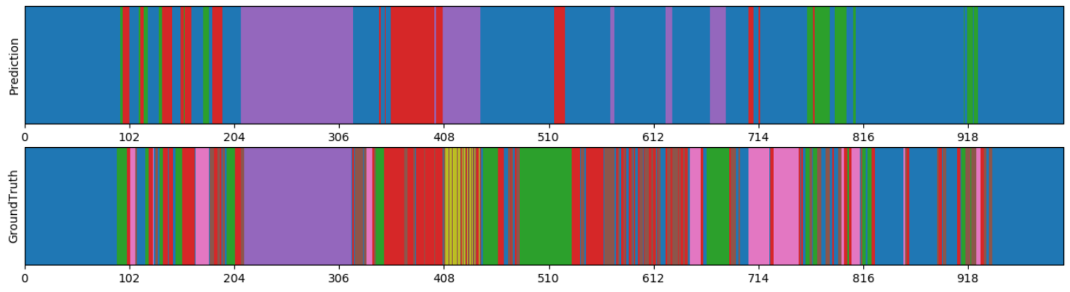
Result:

The accuracy of 7 videos in validation set:

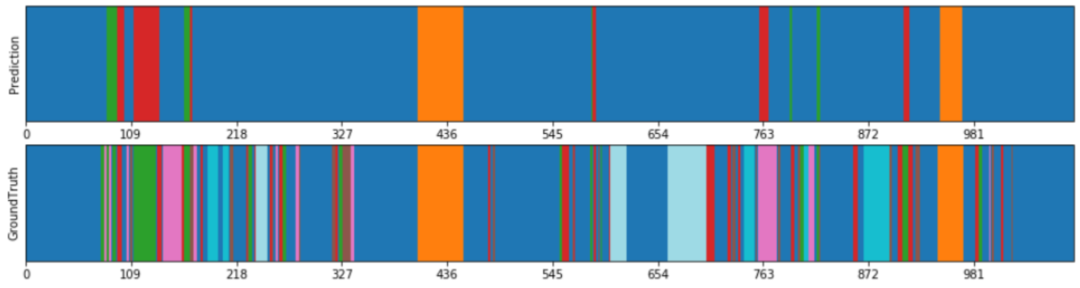
OP01-R02-TurkeySandwich	OP01-R04-Turkey-ContinentalBreakfast	OP01-R07-Pizza	OP03-R04-ContinentalBreakfast
0.4723	0.57128	0.61392	0.53543

OP04-R04-ContinentalBreakfast	OP05-R04-ContinentalBreakfast	OP06-R03-BaconAndEggs
0.64976	0.474683	0.65828

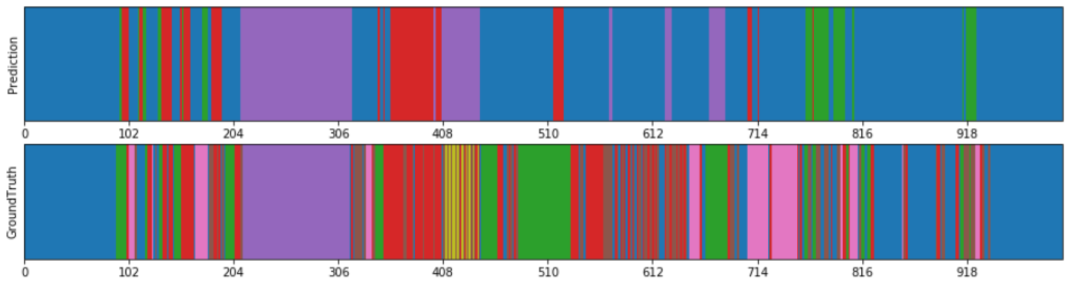
Visualization:
OP06-R03-BaconAndEggs which is the best performance (0.65828) for visualization



OP04-R04-ContinentalBreakfast which is the second best performance (0.64976)



OP01-R02-TurkeySandwich which is the worst (0.4723)



Compare the best and worst case, we can see that the LSTM encoder fails when the action transition happened frequently in short time. We can see that in OP01-R02-TurkeySandwich (worst case), series of actions label are predicted as other (0 in blues color) while if the action is continuous in serious of time (purple color) the LSTM encoder can predict very precision. I think there are two reasons. One is that when we sample from the training set, some information loss in sampling process, the other is the basic characteristic of RNN encoder which uses sequence pattern to predict next sequence pattern. If the sequence pattern features are not clear enough, or change randomly, even the gate mechanism like LSTM, GRU preventing from gradient vanishing still cannot predict the sequence well.