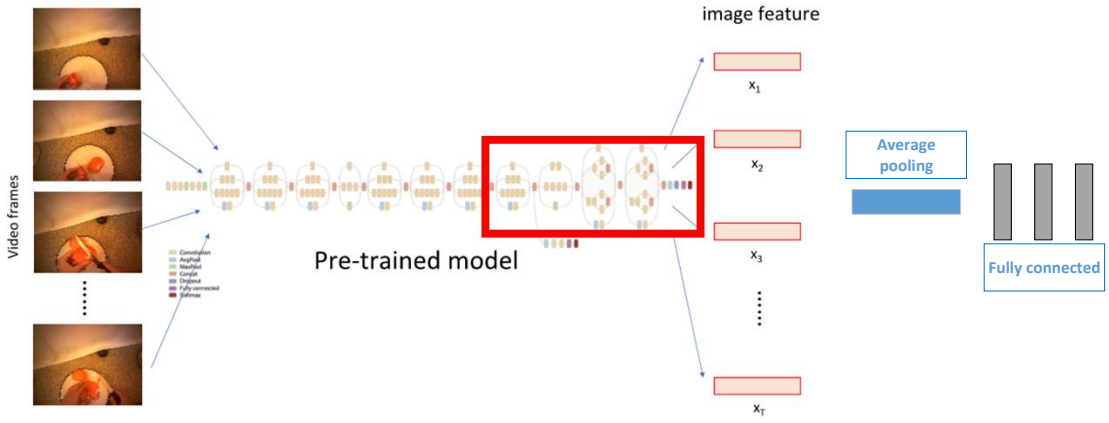
DLCV HW5 廖宜倫 B03901001

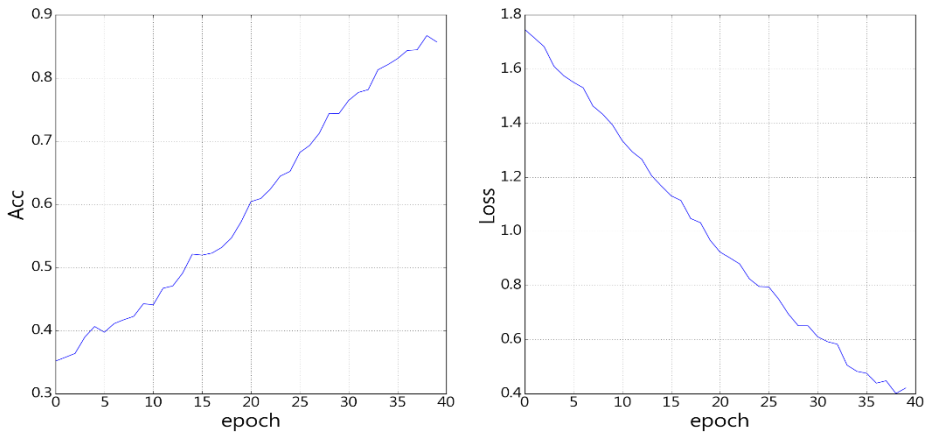
Problem 1:

a. I use pre-trained inception v3 model to extract CNN features and finetune the following layers during training of the task. I use 16 frames of maximum time intervals from videos to extract 16 CNN features and use average pooling to get a 2048-dimensional feature. I input this feature to the fully connected classifier (4096, 4096, 11). I use dropout layers with p=0.2 for fc layers. The activation functions for each fc layer except the last one is ReLU. The learning rate of CNN is 10^{-5}, and that of the classifier is 10^{-3}. I use Adam optimizer and train the network for 40 epochs with batch_size=2.



b.

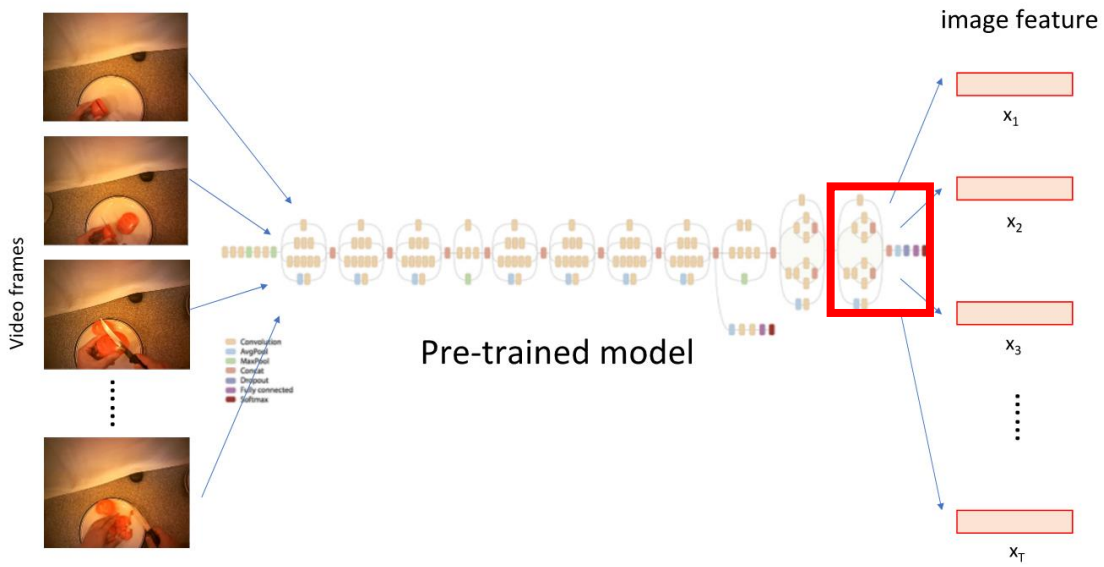
The learning curve is:

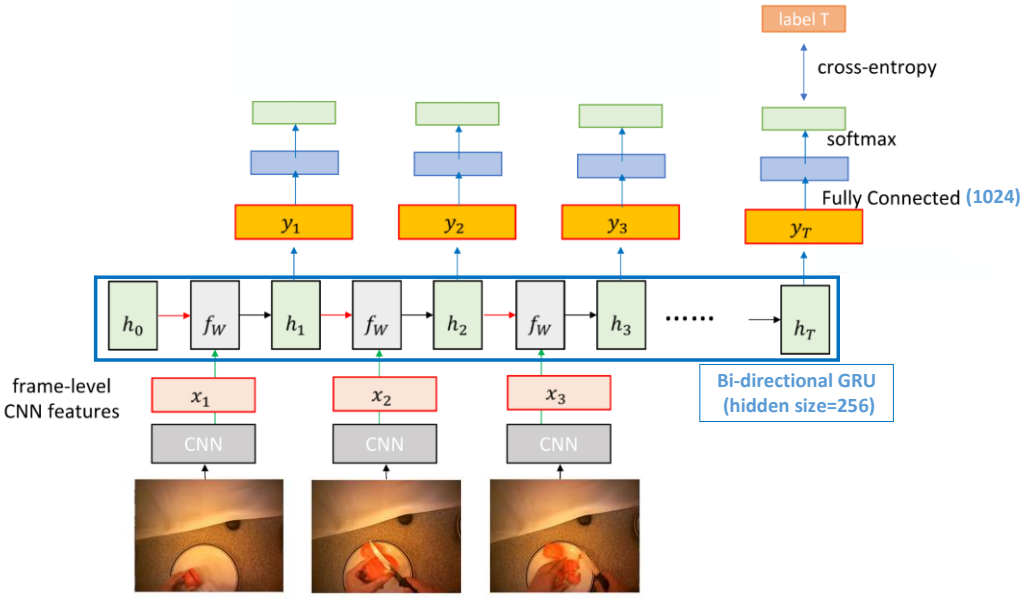


The accuracy on the validation set is about 39.85%.

Problem 2:

a.

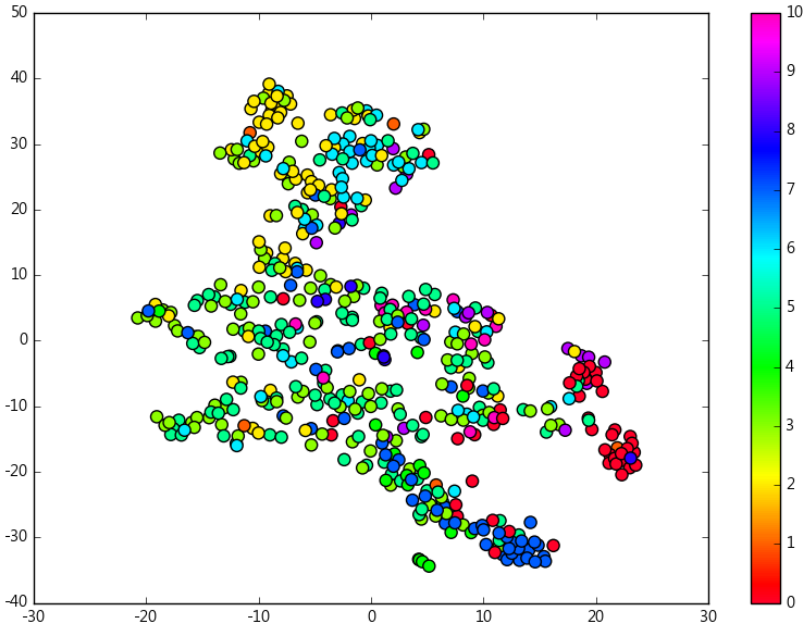




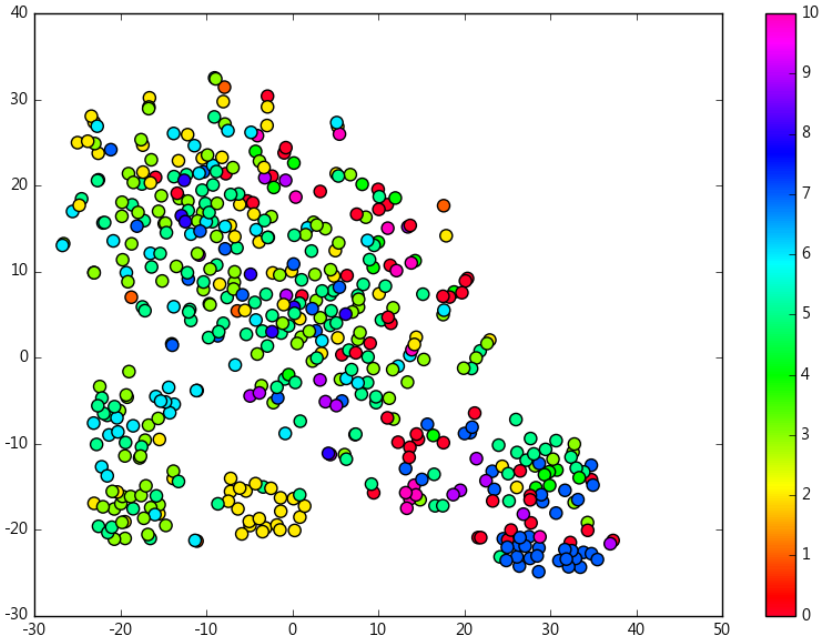
I use pre-trained inception v3 model to extract CNN features and finetune the layers shown in the red box above during training of the task. I input these CNN features to the RNN shown in the above figure. The learning rate of CNN is 10^{-5}, and that of the classifier is 10^{-3}. I use Adam optimizer and train the network for 60 epochs with batch_size=1 without padding.

b. The accuracy on the validation set is about 45.84%.

c. I plot the clustering results of the feature before the last fc layer.



CNN tSNE



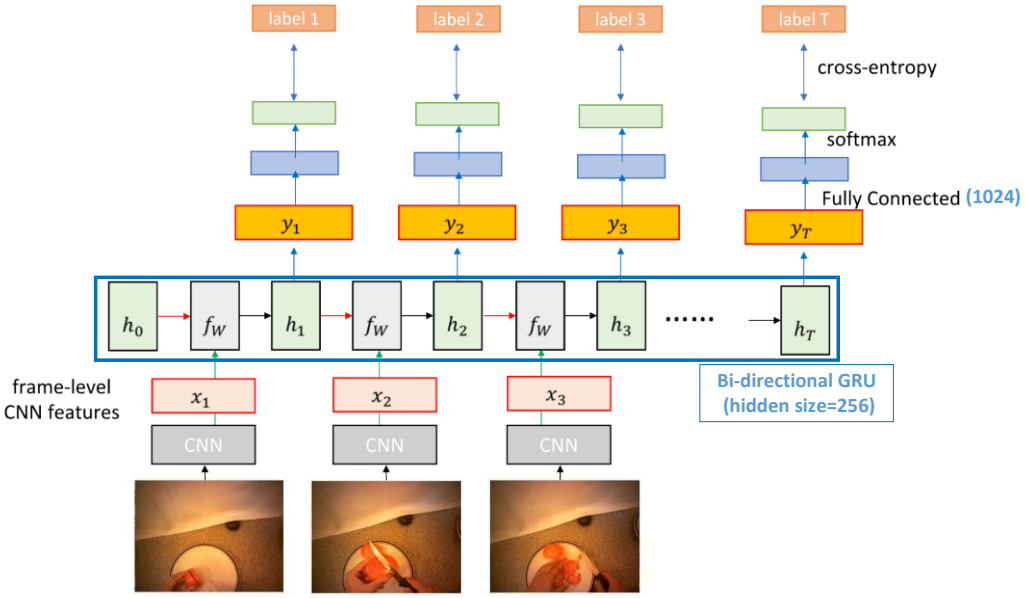
RNN tSNE

From the clustering result of CNN-based features, we can see that only 2 classes (red for 0 and deep blue for 8) are separate from the rest. Since CNN-based features only consider the average of 16 frames, the features of many videos of other classes are mixed and confused. To be specific, points of classes 2, 3, 4 are nearly mixed together. As for RNN-based features, points of class 0 and 8 are still separated. More points of other classes are separated, meaning the distinct features of different classes

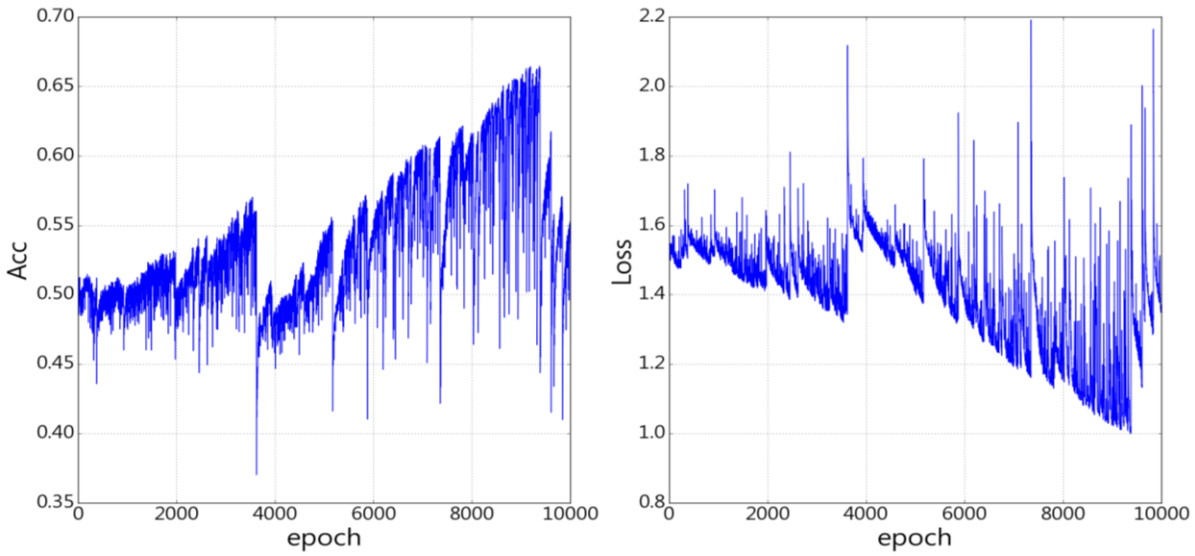
are learned by this network. For example, the bottom left corner suggests that the features of some videos of classes 2, 4, 6 are well recognized and differentiated.

Problem 3:

a. I use the same architecture as Problem 2 and use the model trained for that problem as initialization. I tried Truncated Backpropagation Through Time, but it results in convergence of constant prediction of one class. Different from Problem 2, I do not finetune CNN here but only train the RNN with features of frames. I update the RNN once after comparing predictions of all frames and ground truth labels. This can result in only 29 updates per epoch, and I train this RNN network for 10,000 epochs.

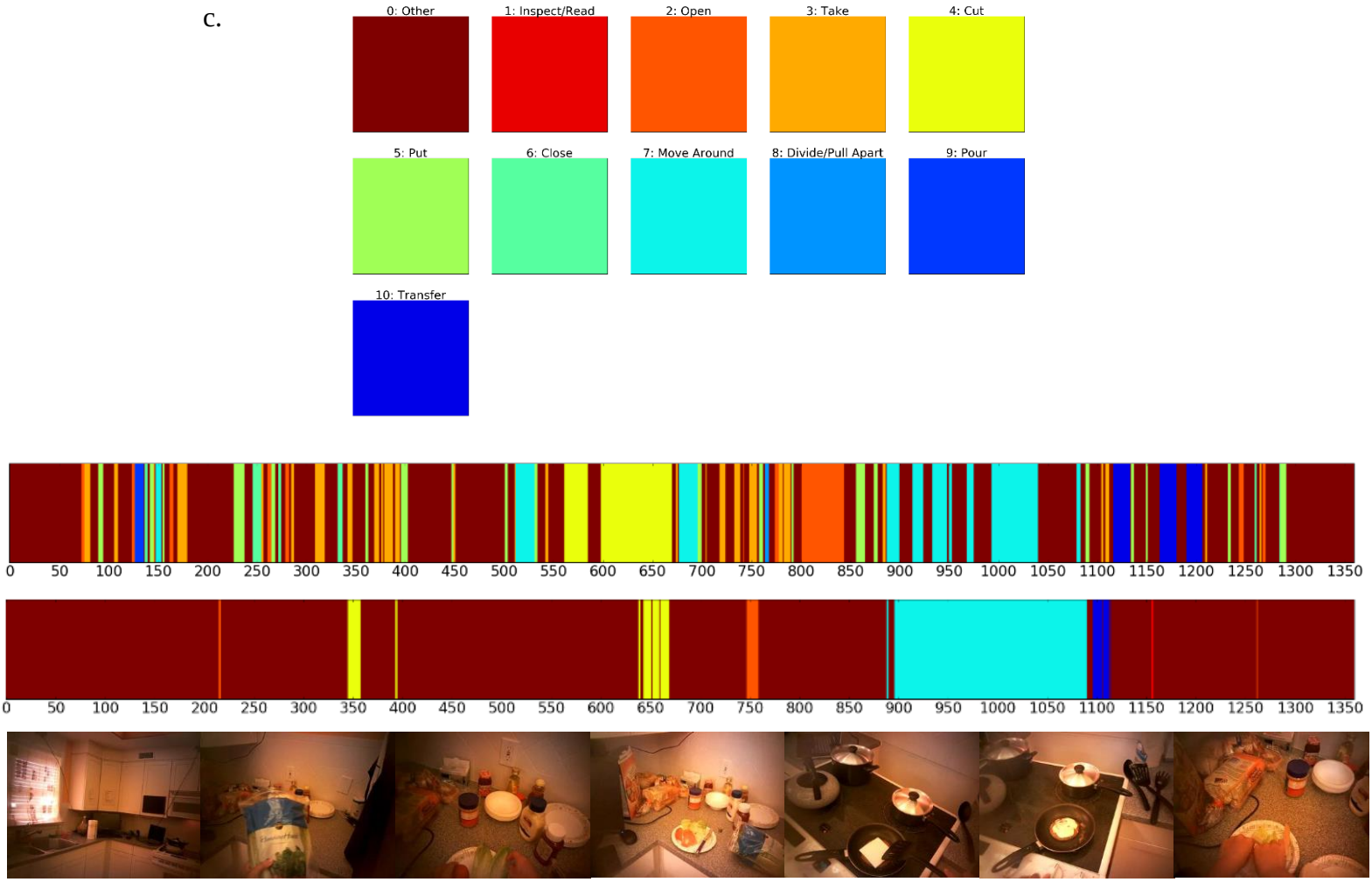


b.



OP01-R03-BaconAndEggs accuracy:	29.07%
OP02-R04-ContinentalBreakfast accuracy:	56.72%
OP03-R02-TurkeySandwich accuracy:	36.99%
OP05-R07-Pizza accuracy:	34.61%
OP06-R05-Cheeseburger accuracy:	55.00%
Total Accuracy:	40.94%

c.



I plot all the labels and predictions for the last full video, OP06-R05-Cheeseburger. There are 1360 frames in this video. As for frames, I plot those with index *200i* (*i*=0~6). From the visualization result, I find that the network tend to predict labels of actions that occupy dominant time intervals. For example, for frames 0~500, actions except “Other” occupy short intervals of time. As RNN produces outputs with consideration of sequences of input data, I postulate that it requires certain long sequences of data to change the output of this network. Since lots of input data are not long enough to change the output of RNN, the network thus keeps predicting what it

has predicted on the last step, possibly the one with relatively long intervals.