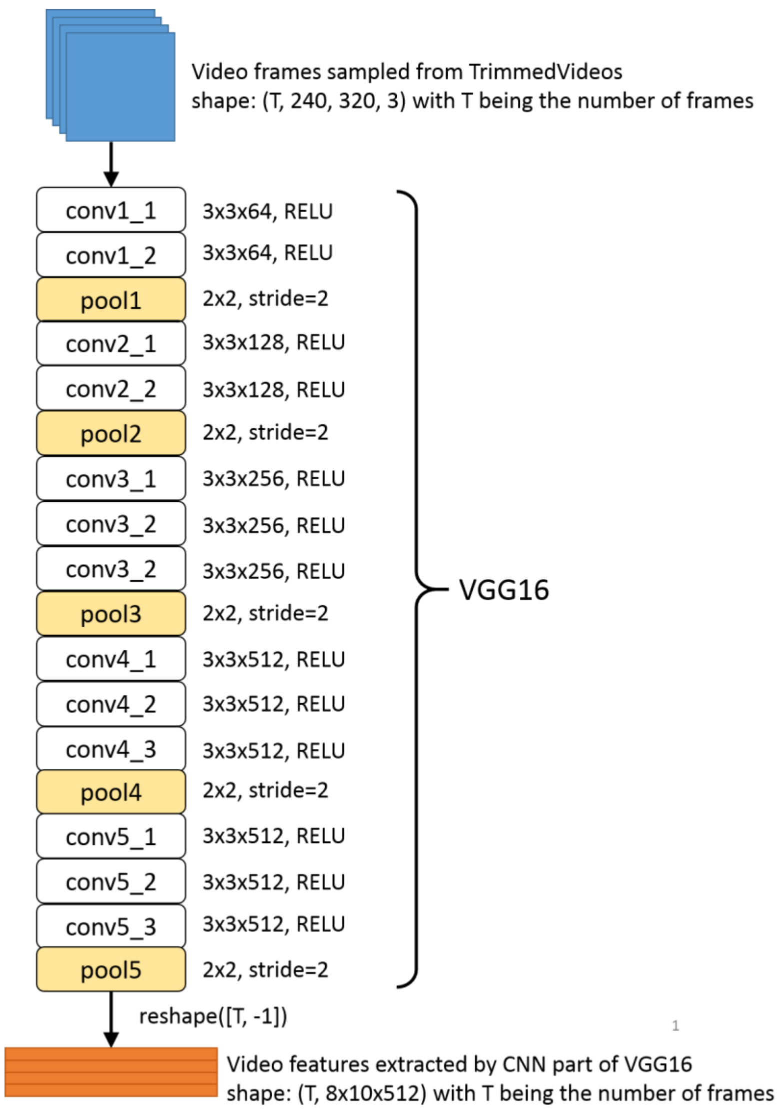
Name: 林家慶 Dep.: 電機博二 Student ID: D05921018

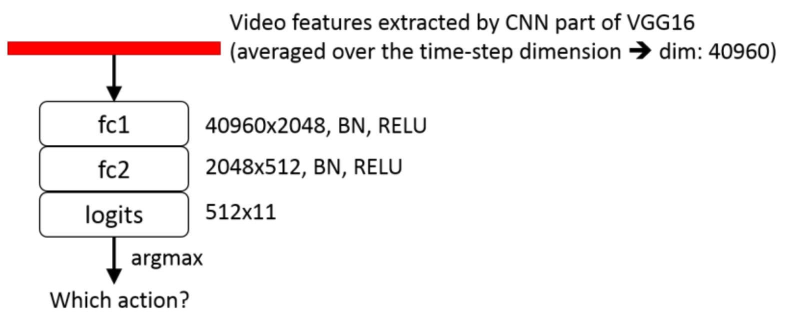
[Problem1]

1. **(5%) Describe your strategies of extracting CNN-based video features, training the model and other implementation details.**

   I used the readShortVideo() and getVideoList() functions in reader.py provided by TA to sample frames from each video, and then used the CNN part of VGG16 to extract CNN-based features (without fine-tuning), as illustrated in the following figure:



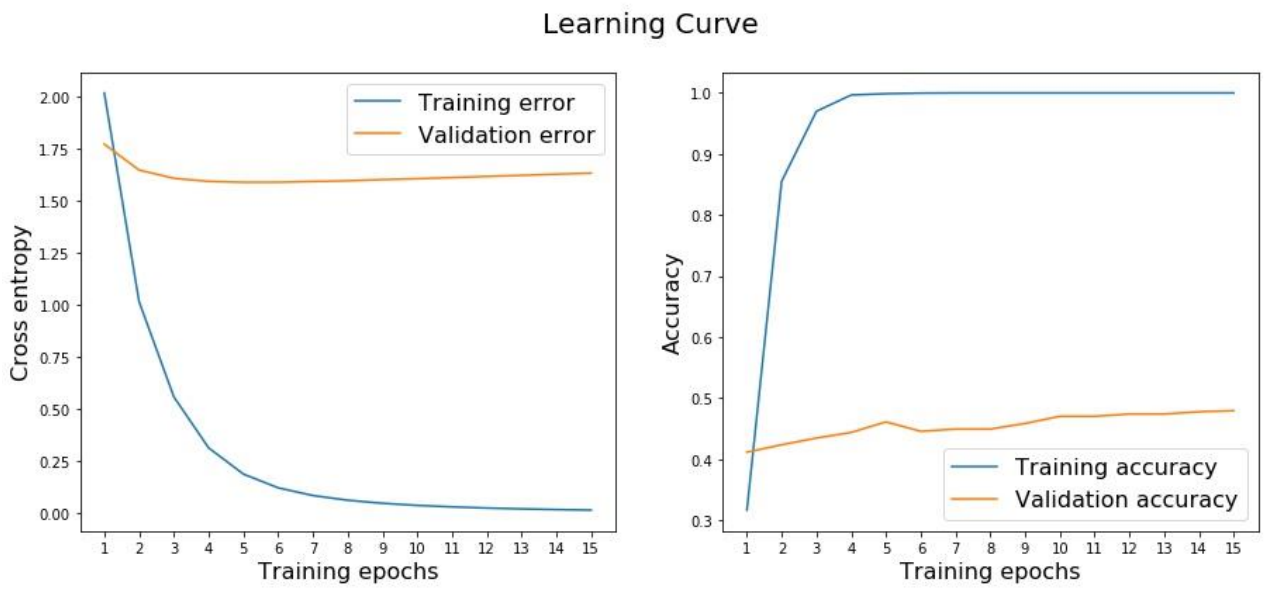
   To handle the issue of different number of frames, I simply averaged all CNN-based features over the time-step dimension to form a single 40960-D vector for each video. Then, I built a DNN to train the action classifier, as illustrated in the following figure:


Video features extracted by CNN part of VGG16
(averaged over the time-step dimension → dim: 40960)
fc1
40960x2048, BN, RELU
fc2
2048x512, BN, RELU
logits
512x11
argmax
Which action?

I used Adam optimizer with *momentum*=0.5 and initial *learning_rate*=5e-5 to minimize the training loss, which is computed by the tf.nn.softmax_cross_entropy_with_logits() function. The batch size is set to 32 and the maximum number of epochs is set to 50. I also implemented a simple early-stopping mechanism by monitoring validation loss with patience 10 and kept the best model with the lowest validation loss. The accuracy on the validation set reach 0.45 after 10 epochs.

2. **(15%) Report your video recognition performance using CNN-based video features and plot the learning curve of your model.**

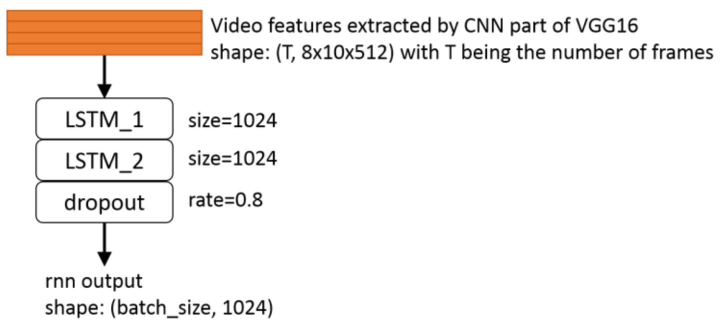
   The best model was obtained after the 5th training epoch with validation loss 1.588220 and validation accuracy 0.461029.



[Problem2]

1. **(5%) Describe your RNN models and implementation details for action recognition.**

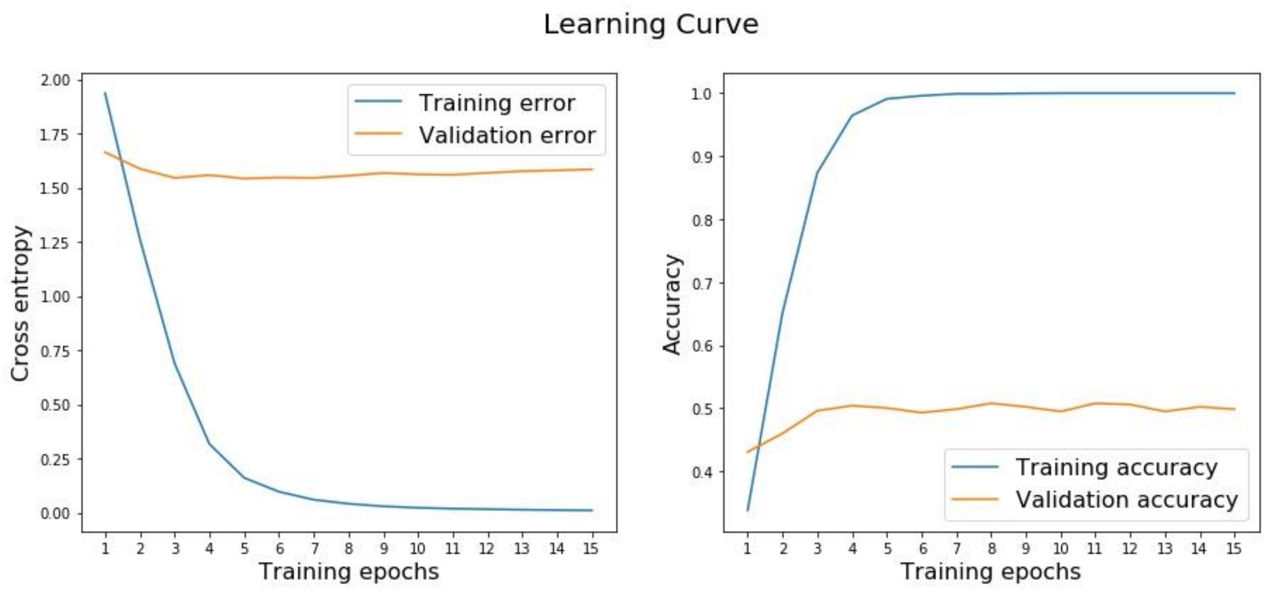
   I used the features extracted and saved in **Problem 1** and built a LSTM netwrok to train the action classifier. With functions BasicLSTMCell() and MultiRNNCell() in the tensorflow.contrib.rnn package, I used a LSTM cell with 2 layers, both have hidden dimension 1024, as illustrated in the following figure:


Video features extracted by CNN part of VGG16
shape: (T, 8x10x512) with T being the number of frames
LSTM_1
size=1024
LSTM_2
size=1024
dropout
rate=0.8
rnn output
shape: (batch_size, 1024)

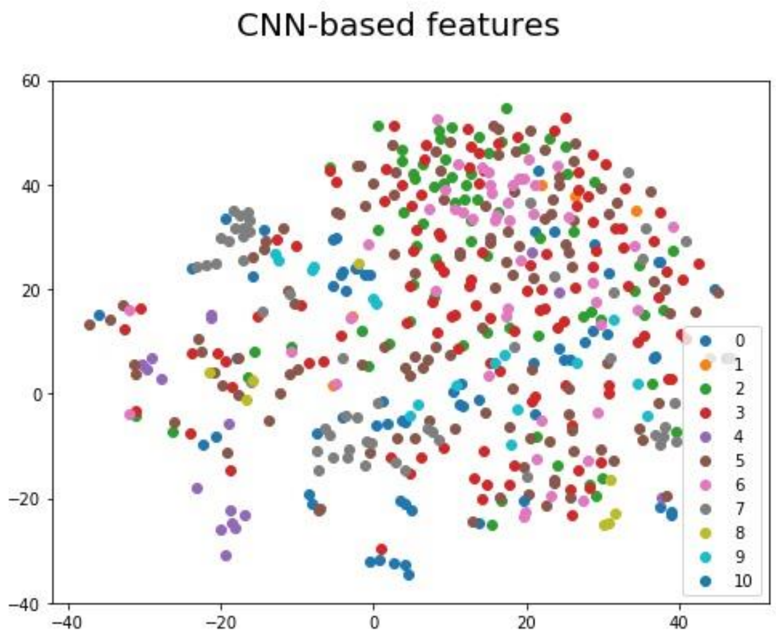
To handle the issue of different number of frames, I set a variable *max_seq_len*=25 to restricted the maximum allowed number of frames of each video. If a video has less than 25 frames, I added zero paddings; Otherwise, if a video has more than 25 frames, I just evenly sampled 25 frames among those frames. The "valid" number of frames for each video is recorded.

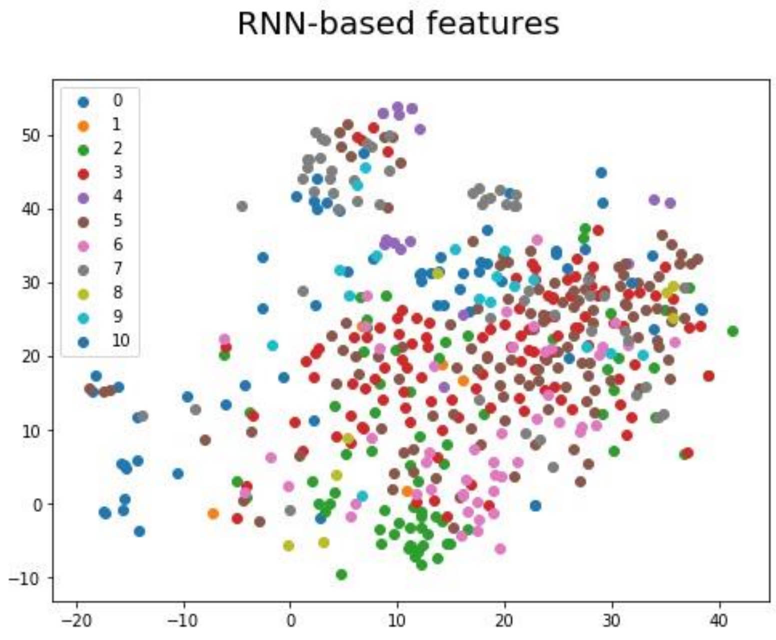
To excute the LSTM operation, I used the tf.nn.dynamic_rnn() function and use the aforementioned "valid" number of frames to extract the correct output for each video. Hence, for each video, there will be a 1024-D rnn output vector, which is then fed into a DNN with one hidden layer (512-D) to get the final output (11-D logits).

I used Adam optimizer with *momentum*=0.5 and initial *learning_rate*=5e-5 to minimize the training loss, which is computed by the tf.nn.softmax_cross_entropy_with_logits() function. The batch size is set to 32 and the maximum number of epochs is set to 50. I also implemented a simple early-stopping mechanism by monitoring validation loss with patience 10 and kept the best model with the lowest validation loss. The best model was obtained after the 5th training epoch with validation loss 1.541819 and validation accuracy 0.500368. The learning curve is as follows:



2. **(15%) Visualize CNN-based video features and RNN-based video features to 2D space (with tSNE). You need to generate two separate graphs and color them with respect to different action labels. Do you see any improvement for action recognition? Please explain your observation.**





As can be seen from the above two figures, CNN-based features of different actions spread out more widely in the 2-D plane, with dots in **Green** (action 2: *Open*) have a trend toward the upper-right corner, and dots in **Purple** (action 4: *Cut*) have a trend toward the lower-left corner. On the other hand, RNN-based features are more concentrated within each group. For example, dots in **Green** (action 2: *Open*) form a very compact group in the lower region, dots in **Brown** (action 5: *Put*) form a group in the right region, and dots in **Grey** (Action 7: *Move Around*) form two small groups in the upper region.

The difference may be due to the much higher dimension of the original CNN-based features (8*10*512 = 40960) compared to the RNN-based features (1024).

[Problem3]

1. **(5%) Describe any extension of your RNN models, training tricks, and post-processing techniques you used for temporal action segmentation.**

   The model structure and most of the implementation details are the same as **Problem 2**, except that: (1) the LSTM cell now has 512 hidden dimensions (instead of 1024) for both layers; (2) the maximum allowed number of frames is extended to 350 (instead of 25); and (3) all 350 outputs are used to predict the actions for all input frames.

2. **(10%) Report validation accuracy and plot the learning curve.**

3. **(10%) Choose one video from the 5 validation videos to visualize the best prediction result in comparison with the ground-truth scores in your report. Please make your figure clear and explain your visualization results. You need to plot at least 300 continuous frames (2.5 mins).**

[BONUS]