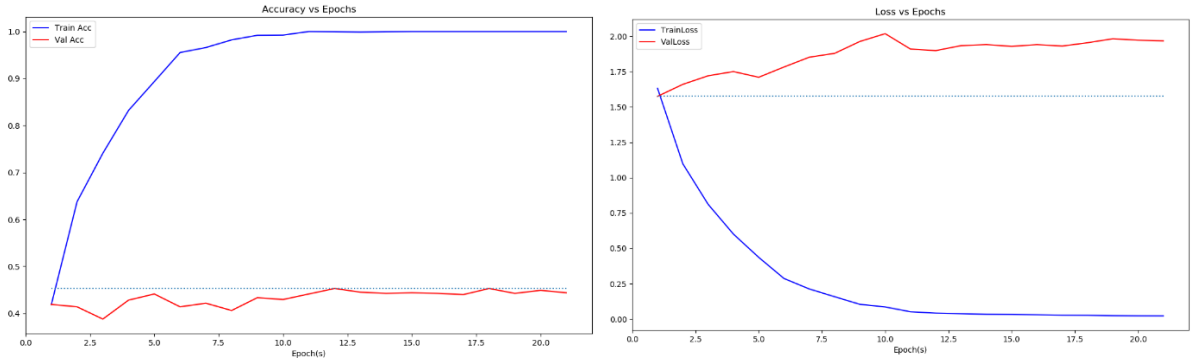
Name: 李華健 Dep.: 電機三 Student ID:B05901119

Problem 1 Data preprocessing:

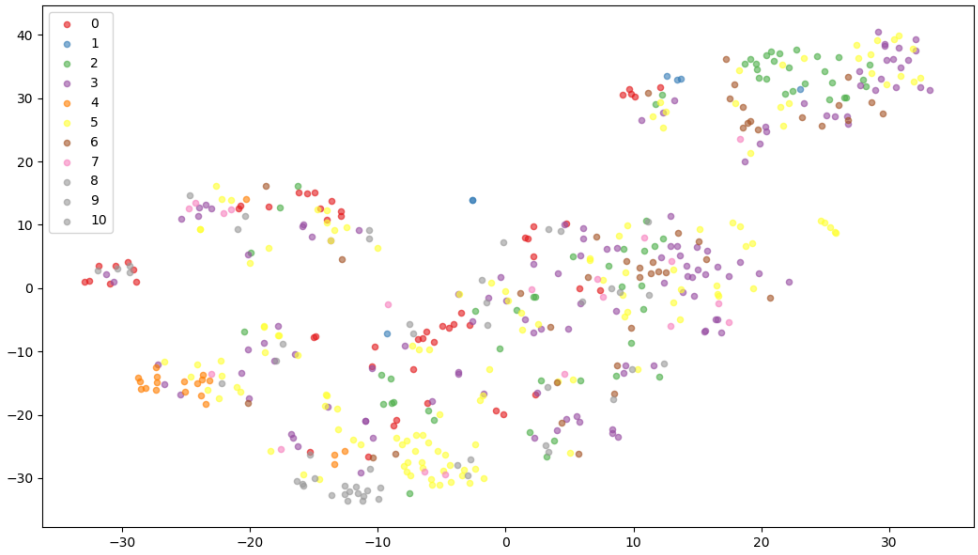
1. Model Performance:
   I choose to save the parameters at epoch 20. In training, the training accuracy reached 100%, and the validation accuracy is 44.9%



[Left: Accuracy curve of training(blue) and validation(brown)]
[Right: Loss curve of training(blue) and validation(brown)]

2. Reproduced Validation Accuracy:
   44.6034%

3. T-SNE in validation dataset:



4. Strategies abstract:
   Sample 4 images from the short video, passed through a resnet50 model, and stored as an 8192-dimension feature vector. Then passed through a classifier network, which has 2 layers of fully connected, the output vector is a dimension of 11 prediction result.

5. Training details and model structure

   I. CNN:

   Use resnet50 pretrained model in PyTorch, with the mean/std shifting. Truncate the network after the average pooling, and adjust the kernel size of the average polling so that the model is fed in (3, 240, 320) and generate (2048) vector.

   II. Classifier:

   Linear(8192, 2048)
   BatchNorm1D(2048)
   ReLU()
   Dropout(0.5)
   Linear(2048, 11)

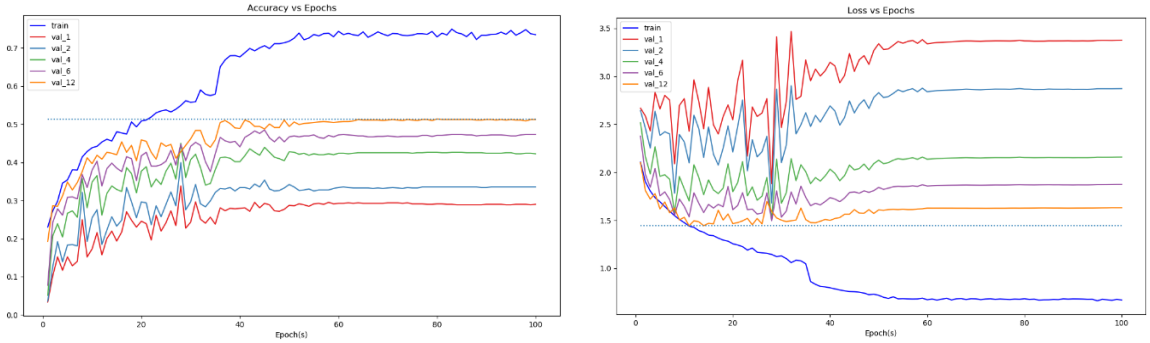
   III. Train setting:

   I have chosen Adam as optimizer, initial learning rate 1e-4, decay to 1e-5 after epoch 10. The Regularization was set as 1e-4. The loss function is Cross-entropy. Batch size 16, epoch was set as 50, and stopped at 20 because of very good performance on training set.

Problem 2 Trimmed action recognition:

1. Model Performance:

   I choose to save the parameters at epoch 100. When the training was done, the training accuracy is 73.50%, and the validation accuracy is 51.24%. When training, I have chosen down-sample rate 12 at training, and monitoring [1, 2, 4, 6, 12] at validation. 12 preforms best in my experiments.

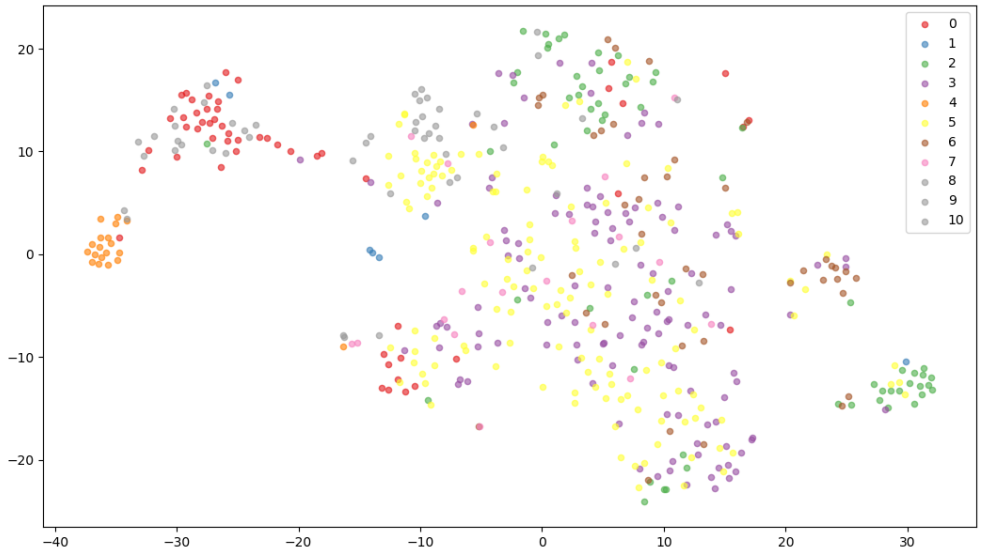


[Left: Accuracy curve of training(blue) and validation (val_12: orange)]
[Right: Loss curve of training(blue) and validation (val_12: orange)]

2. Reproduced Validation Accuracy:

   50.8453%

3. T-SNE in validation set



Compare with the t-SNE of problem 1, I saw that longer distance or significant separating of each class in here. Such as class ‘4’ at the leftmost of the figure and class ‘5’ represented by yellow points. Seems the RNN mechanism can provided an improve of classification accuracy.

4. Strategies abstract:
Down-sampling 12 from the short video, passed through a resnet50 model, and stored as multiple 2048-dimension feature vectors. Then passed through a 2-layer LSTM network with the 128-dimension hidden states. After all feature vectors is fed in LSTM, take the hidden state vector and passed a 2-layer classifier to generate the class prediction result.

5. Training details and model structure
   I. CNN: Same as Problem1 (ResNet50), skipped here.

   II. Recurrent Network:
   LSTM(input=2048, hidden=128, dropout=0.2, layer=2)
   Linear(128, 128)
   ReLU()
   Dropout(0.2)
   Linear(128, 11)

   III. Train setting:
   I have chosen SGD as optimizer, initial learning rate 1e-3, adjusted to (1e-4, 1e-5, 1e-6) at epoch 35, 50, 60. The Regularization was set as 1e-4. The loss function is Cross-entropy. Epoch was set as 100 and batch size is 1. For the weight initialization, I have used **orthogonal weight initialized** and set the **bias of the forget gate as 0**.

Problem 3 Temporal action segmentation:

1. Model Performance:
   The model performs bad on this dataset, one of the reasons is the model can't handle the ununiform training data, where the label "Other" occurs very often. And the model would rather predict a '0' to get the lower loss.

   My post-process strategy is filling the blank between 2 action labels if they are near enough (such as 1 second, or 24 frames). But it doesn't work because of the poor learning status.

2. Reproduced Validation Accuracy:

3. Predict result visualization (Case: OP01-R02-TurkeySandwich)

   The first row of the color bar is the ground truth, second one is the predicted result, without post-processing, and the last one is the predicted result with the highest confidence except case '0'.

   In this experiment, we can find that the model always throws a '0' to get a higher score. And the 3rd row shows that one case takes the highest predicted confidence except '0'.

[Predicts result of the validation video]

   To improve the model, I think a better learning schedule is needed. Such as trimmed the videos, augment the action labels. Try to use TBPTT.

4. Strategies abstract:
   Down-sampling 4 from the short video, passed through a resnet50 model, and stored as multiple 2048-dimension feature vectors. Then passed through a 2-layer LSTM network with the 512-dimension hidden states. Predict the labels per frame by taking the hidden state vector to passed a 2-layer classifier to generate the class prediction result.

5. Training details and model structure
   I. CNN: Same as Problem1, 2(ResNet50), skipped here.

   II. Recurrent Network:
   LSTM(input=2048, hidden=512, dropout=0.2, layer=2)
   Linear(512, 512)

ReLU()
Dropout(0.2)
Linear(512, 11)

III. Train setting:
Fine-tuned the model parameter trained at problem 2. Because of the dimension and different dimension, first I have train the model with the above setting on problem 2.

I have chosen SGD as optimizer, initial learning rate 1e-4. The Regularization was set as 1e-4. The loss function is Cross-entropy. Epoch was set as 1000.

Trimmed the training data and TBPTT is tried on my experiments. Usually random sample 64 / 128 frames from a video