Deep Learning for Computer Vision HW#4

B05901182 電機四 潘彥銘

Problem 1: (20%)

Collaboration: B05901027 詹書愷、B05602042 林奕廷

Reference: None

1. (5%)

Each video, I randomly chose two frames regardless the order. Then fed them into the feature extractor(Pretrained Resnet-50) and concatenated with each other. Till now, the dimension is (Batch size, 4096).

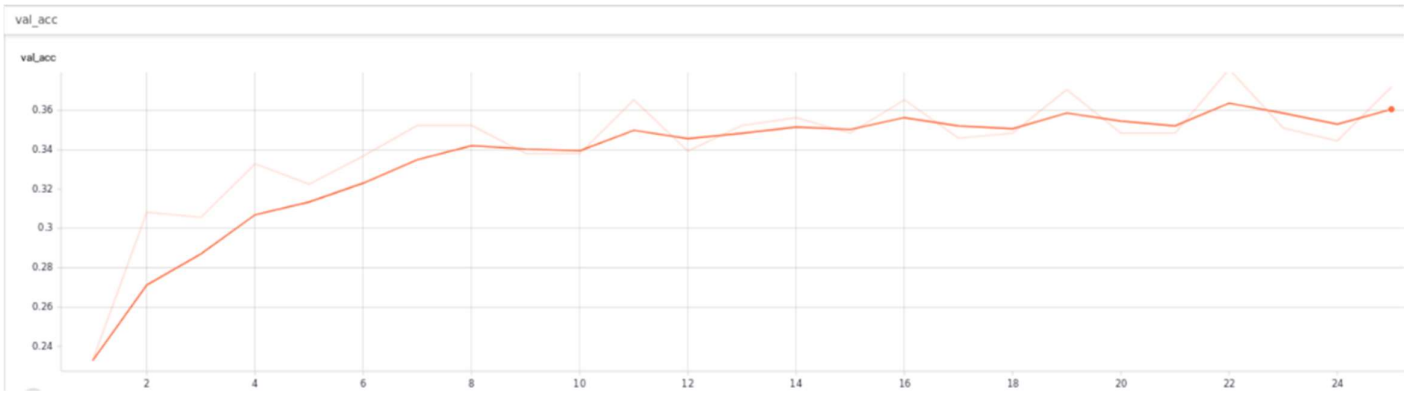
And I fed them into the classifier whose architecture is like the figure below:

```
class Classifier(nn.Module):
    def __init__(self):
        super(Classifier, self).__init__()
        self.clf = nn.Sequential(
            nn.Linear(4096, 2048),
            nn.ReLU(),
            nn.Linear(2048, 512),
            nn.ReLU(),
            nn.Linear(512, 11)
        )
```

The output of the classifier is the probability of prediction of each class. What's more, I chose cross-entropy loss and adam optimizer to train my classifier. The learning rate is 0.0002 with step scheduler(learning rate will times 0.8 each epoch)

The figure below is the loss curve of training set:

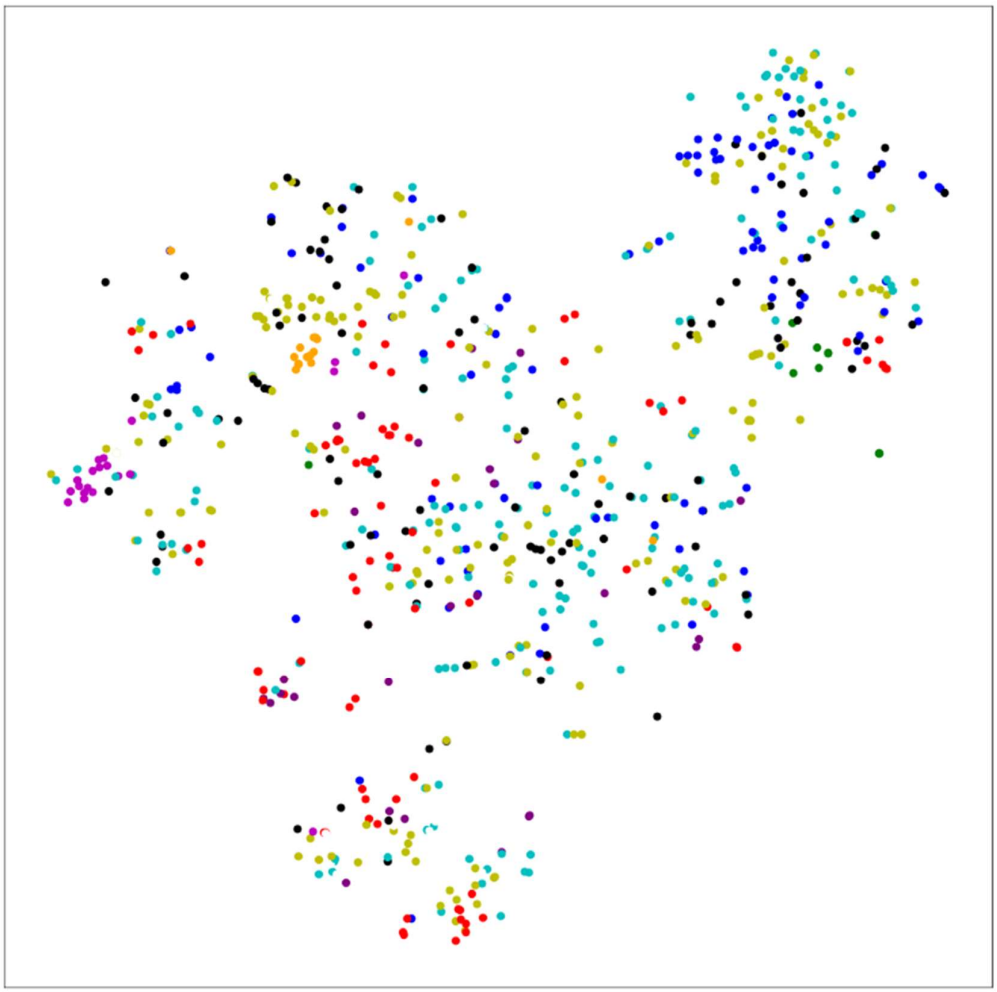
The figure below is the loss curve of validation set



2. (5%)

Video recognition performance (valid): 35.24%

3. (10%)



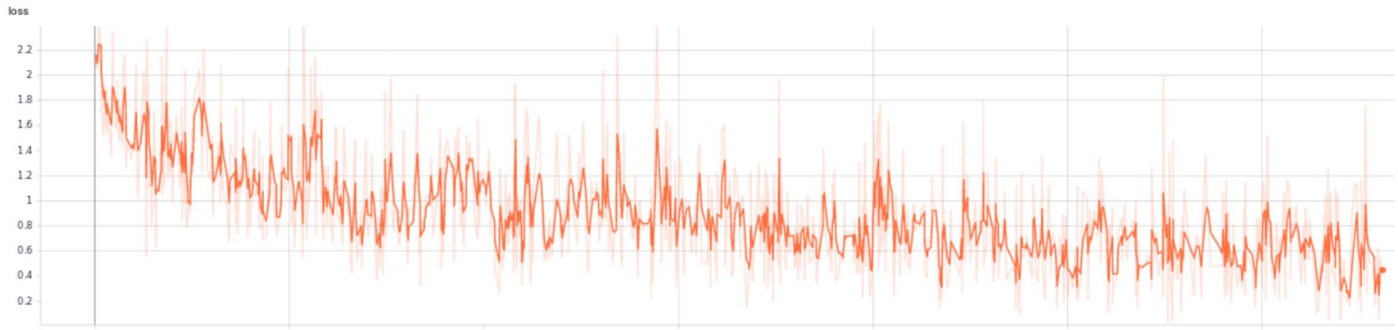
Problem 2: (40%)
Collaboration: B05901027 詹書愷
Reference:
1. https://zhuanlan.zhihu.com/p/59772104

1. (5%)
For each video, due to the memory limitation, I chose 10 frames only. Then fed them into the feature extractor(Pretrained Resnet-50). I will store the features in each batch_size time. And then fed them into the GRU based rnn model. I chose the last time step feature and fed into two fully-connected layer classifier, whose structure is like the figure below:

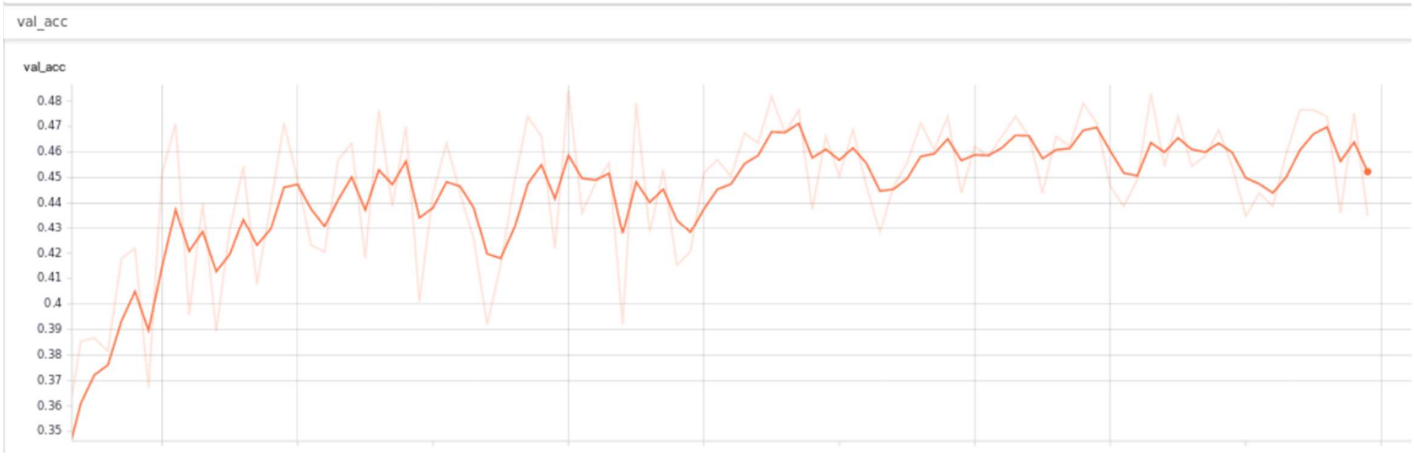
```
class RNNClassifier(nn.Module): # gru with hidden size 256
    def __init__(self, args):
        super(RNNClassifier, self).__init__()
        self.gru = nn.GRU(
            input_size = 2048,
            hidden_size = args.hidden_size,
            num_layers = args.num_layer,
            batch_first = True
            )
        self.clf = nn.Sequential(
            nn.Linear(256, 64),
            nn.ReLU(),
            nn.Linear(64, 11)
        )
        self.batch_size = args.clf_batch
        self.hidden_size = args.hidden_size
```

I chose cross-entropy loss and SGD optimizer with lr=0.01 and momentum=0.9

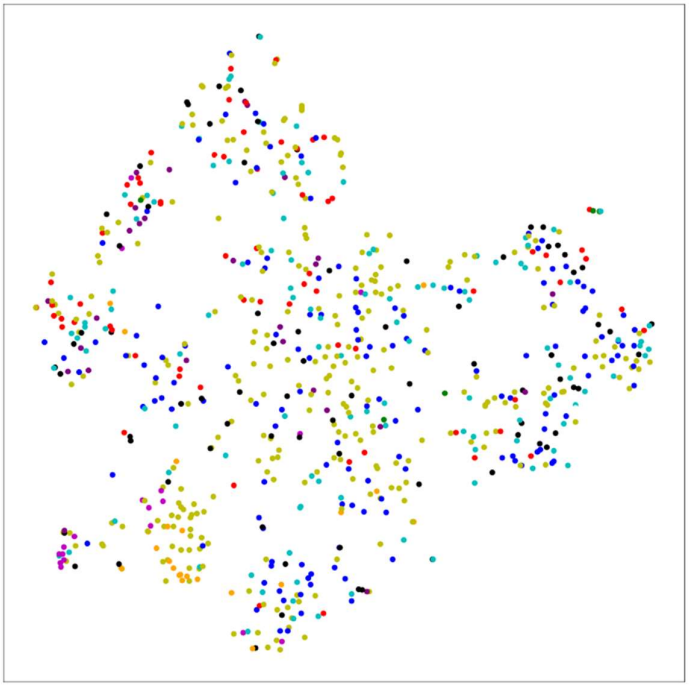
The figure below is the loss curve of training:



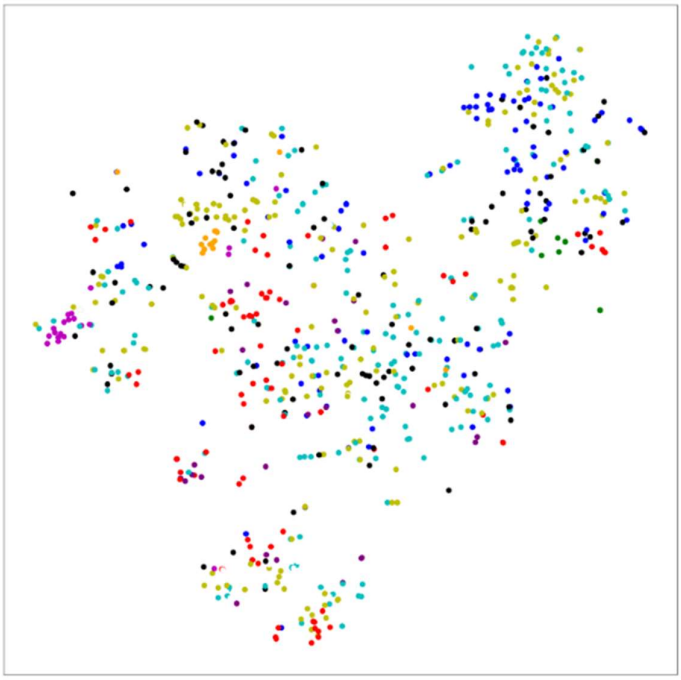
The figure below is the loss curve of validation set:



2. (10%)



RNN-based video features



CNN-based video features

In my observation, I think there is a little improvement. The points with same color are getting closer in the left figure, compared to the right figure, though it's not that obvious. I think it's because my RNN-based model's accuracy is not high enough(47.98%). The accuracy of my CNN-based model is about 35%. There is only about 13% gap.

Problem 3: (40%)
Collaboration: B05901027 詹書愷
Reference:

1.(5%)

For each video, I chose 256 frames due to the memory limitation. And I use the same model, optimizer and loss function in problem 2. The difference is that this time I chose each time step (256 time steps in total) and fed them into the classifier.

2. (20%)

Validation Video	Accuracy
OP01-R02-TurkeySandwich	490/1012 (48.42%)
OP01-R04-ContinentalBreakfast	598/982 (60.90%)
OP01-R07-Pizza	1585/2471 (64.14%)
OP03-R04-ContinentalBreakfast	485/889 (54.56%)
OP04-R04-ContinentalBreakfast	668/1085 (61.57%)
OP05-R04-ContinentalBreakfast	524/948 (55.27%)
OP06-R03-BaconAndEggs	1018/1551 (65.64%)

3. (15%)

I use the frames from OP04-R04-ContinentalBreakfast to generate the image above. The upper color bar is the prediction and the lower color bar is the ground truth label. The different color represents different action label. And the middle bar is the frames from the video.