Name: 蔡昕宇 Dep.:電機三 Student ID:B04901096

[Problem1]

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

   在 preprocess 的部分,將影片 down sample 為 2fps，使得運算的時間減少，且將 frames normalize，其 normalize 的方法與 ImageNet[1]的 preprocess 中使用的方法相同，如下圖所示

```
transforms.Normalize(mean=[0.485, 0.456, 0.406],
                     std=[0.229, 0.224, 0.225])
```

   通過 pretrained model 之後 average pooling，再用來當作 CNN feature input 來 train。在實作中嘗試過 Vgg19, DenseNet121, Resnet50。但後來使用 Resnet50 作為最後實作的 pretrained feature model。其中實作中沒有 fine tune pretrained model 的參數，直接 train fully connected 的部分。架構如下

```
CNN(
  (fc): Classfier(
    (fc): Sequential(
      (0): Linear(in_features=2048, out_features=4096, bias=True)
      (1): BatchNorm1d(4096, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace)
      (3): Linear(in_features=4096, out_features=1024, bias=True)
      (4): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (5): ReLU(inplace)
      (6): Linear(in_features=1024, out_features=11, bias=True)
    )
    (output): Softmax()
  )
)
```

   經過 Resnet50 model 之後，取其倒數 average pooling 之後的 output，其 output size 為 (batch size, 2048, 1, 1)，然後將 fully connected 的 layer 換成上述所示的架構。

   其他 optimizer 與 criterion 的參數如下：

   Optimizer: Adam (lr = 0.0001)

   Criterion: Cross Entropy Loss

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

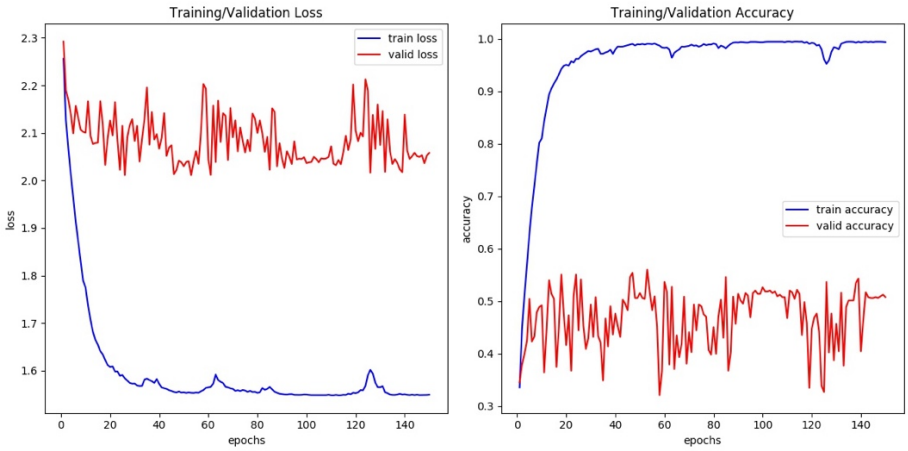
   使用 Resnet50 作為 pretrained model，再接新的 fully connected model。結果如下：

   Training accuracy: 0.995

   Validation accuracy: 0.485493

[1] ImageNet example: https://github.com/pytorch/examples/blob/master/imagenet/main.py

Learning curve



可以看到 training loss 在 20 epochs 左右可以下降到 1.6 左右，但是 validation loss 同樣是過了 20 epochs，training loss 大約在 2.0 到 2.2 之間震盪。另外看到 accuracy，training 很快地就可以到接近 1，但 validation accuracy 卻在 0.4 到 0.5 之間震盪，與 loss 同樣的震盪的很劇烈。顯然這樣的 training 似乎 training data 和 validation data 的分佈可能差異較大，才會導致這樣的結果。

[Problem2]

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

Preprocess 的部分與 CNN-based model 相同，皆是將影片 down sample 成 2 fps，再將其通過 Resnet50，也沒有 fine tune pretrained model。而 RNN model 在實作中，架構如下

```
RNN(
  (encoder): Encoder(
    (rnn): LSTM(2048, 512)
  )
  (fc): Classfier(
    (fc): Sequential(
      (0): Linear(in_features=512, out_features=11, bias=True)
    )
    (output): Softmax()
  )
)
```

實作上只有 RNN 的部分使用 LSTM，以下為 LSTM 中架構

input size: 2048
hidden size: 512
layers: 1

沒有使用 bidirectional 的架構，單純使用一層的 LSTM。而與上一題不同的是 fully connected，只使用一層的 Linear layer 作為 classifier。

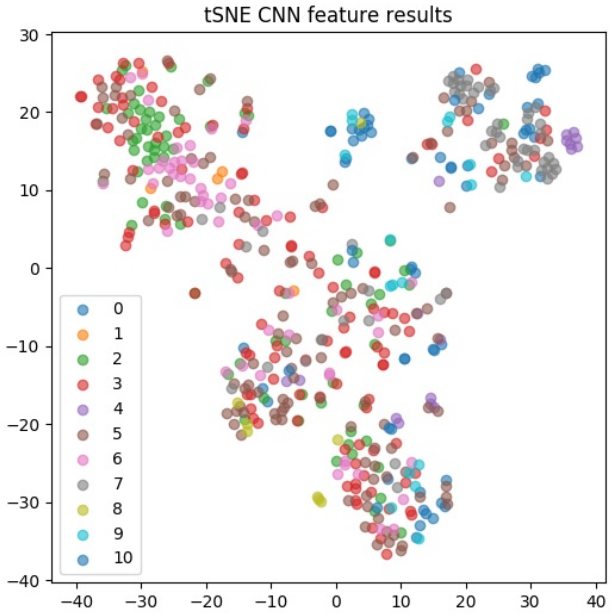
其他 optimizer 與 criterion 的參數如下：

Optimizer: Adam (lr = 0.0001)
Criterion: Cross Entropy Loss

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.

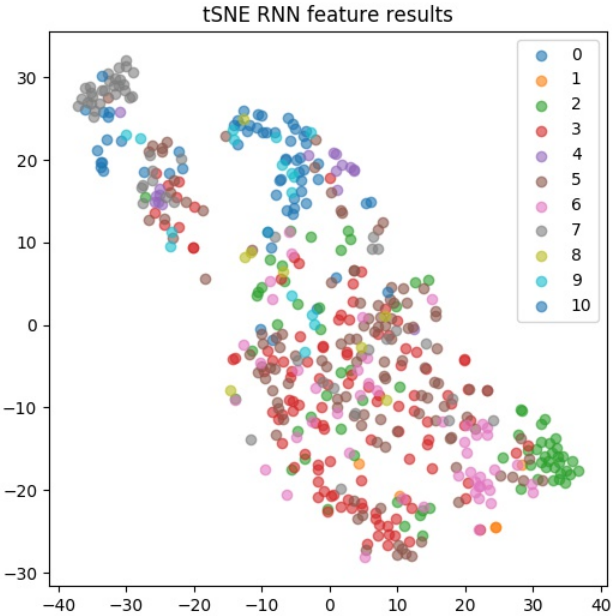
CNN based feature results

是由 Resnet50 出來的所有 frame 作 average pooling，把全部 frame 平均得到的 feature 的結果



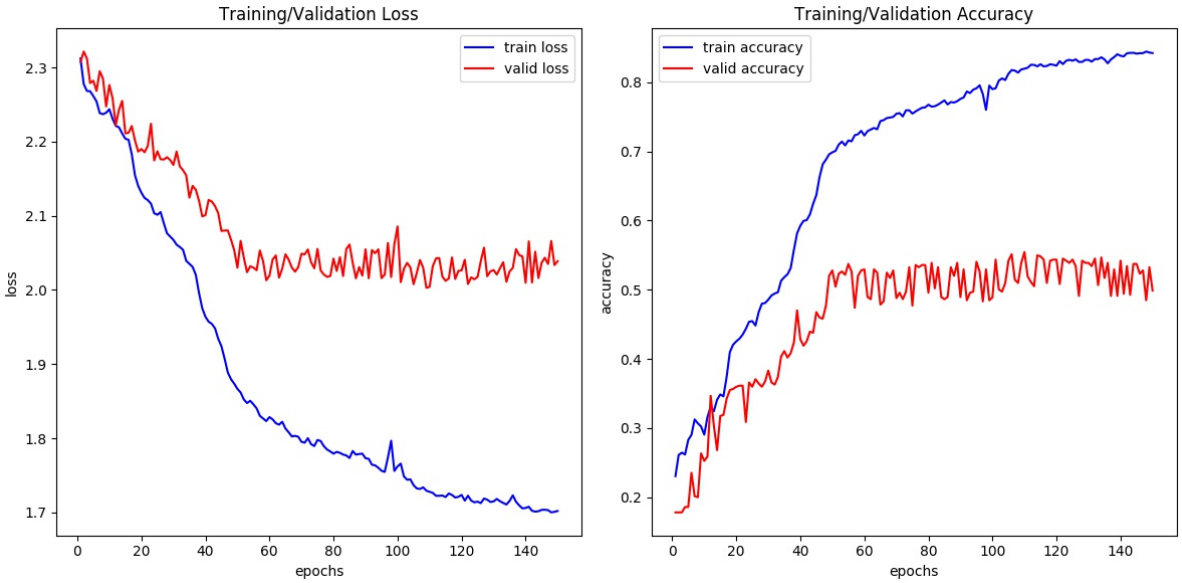
RNN based feature results

是由 Resnet50 出來的所有 frame 通過 LSTM 之後得到的 feature 的結果



這兩者的經過 tSNE 2000 個 iteration 後的結果，總體而言看起來差別好像不大，但就細節的部分來看，RNN 的架構看起來有同一個類別群聚在最外側的現象，然而 CNN-based 的 feature 則比較像是每個類別都混在一起，分不太出來兩兩類別的區隔，換了幾個 random seed 看到差不多的結果。

以下為 RNN-feature 的 learning curve



由兩者的 training loss 來看，收斂的速度相較沒有接而言較慢，可能是因為使用的 fully connected layer 較淺，classifier model capacity 較為小，但為了控制兩者 parameters 差不多才能做比較，因此才使用較為淺的 fully connected layer。而 validation 而言，在 training 的過程中 validation loss 稍微比第一題 CNN-based 的架構好些，classifier accuracy 也上升了一些。

CNN-based validation accuracy: 0.485493

RNN-based validation accuracy: 0.520309

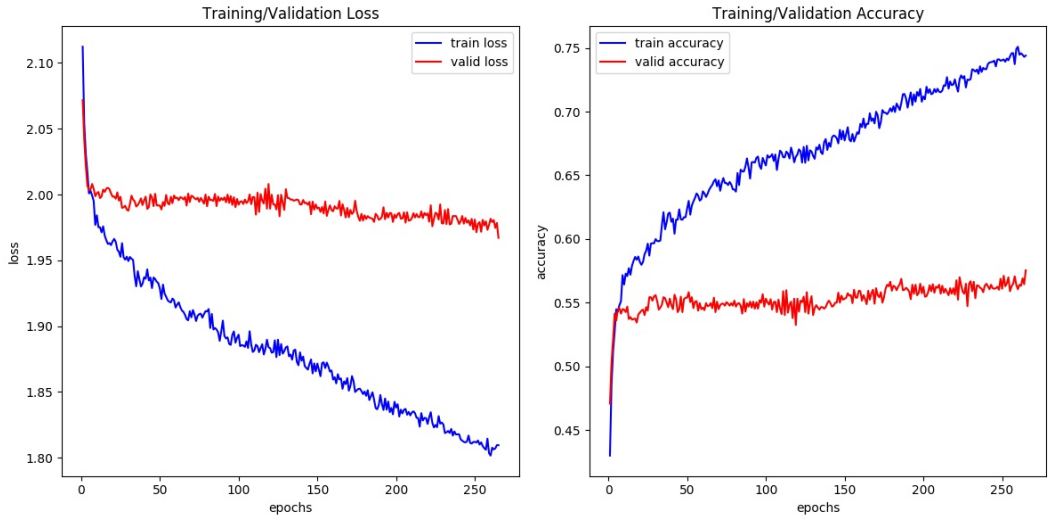
從這樣的結果而言，可能就是因為 RNN-based 的 feature 能夠將同一類別的分在較為靠近的地方，不像 CNN-based 的 feature 則是全部混在一起，使得 classifier 在 RNN-based feature model 有更好的效果。

[Problem3]

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

架構上與第二題相同，此題的架構類似 decoder 的架構。每次 input 一個真實的 frame 生成一個 label，組成 sequence to sequence 的架構。而在這個 task 中，實作中的 video frame 不是每一個 frame 都取。每部影片中，每次都隨機取 512 張 video frame，因此 frame input size 變成 (512, batch size, hidden size),且同一個影片每個 epoch 得到的 frame 也不同，相對來說可以增加 training data。因此實作上將 batch size 設為 8。其他的架構完全與第二題相同。

2. (10%) Report validation accuracy and plot the learning curve.
   Validation accuracy: 3488 / 6104 (≈0.57143)
   Learning curve



可以看到 validation 的 loss 仍然是在 20 epochs 左右會卡在 2 附近，與前幾題相同，accuracy 有略為上升，大致上在 0.55 上下。同樣的這可能也是 training data 與 validation 影片的分佈差異所導致。

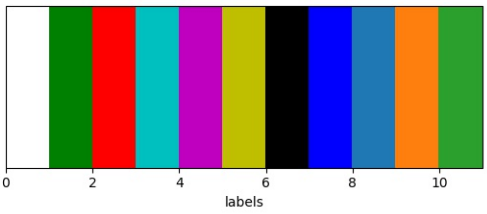
3. (10%) Choose one video from the 5 validation videos to visualize the best prediction result in comparison with the ground-truth scores in your report. Please make your figure clear and explain your visualization results. You need to plot at least 300 continuous frames (2.5 mins).

   五部 validation video accuracy 如下：

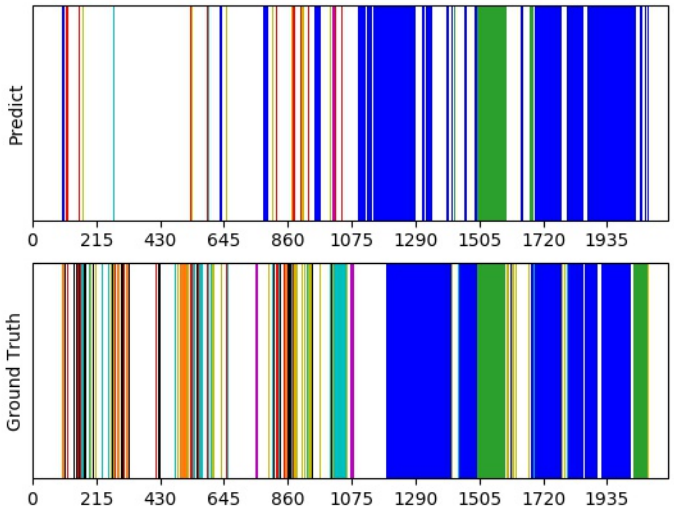
Video Category	Accuracy
OP01-R03-BaconAndEggs	0.59626 (1276 / 2140)
OP02-R04-ContinentalBreakfast	0.62154 (538 / 938)
OP03-R02-TurkeySandwich	0.49475 (424 / 857)
OP05-R07-Pizza	0.54759 (443 / 809)
OP06-R05-CheeseBurger	0.56029 (762 / 1360)

   而總 Validation accuracy: 3488 / 6104 (≈0.57143)

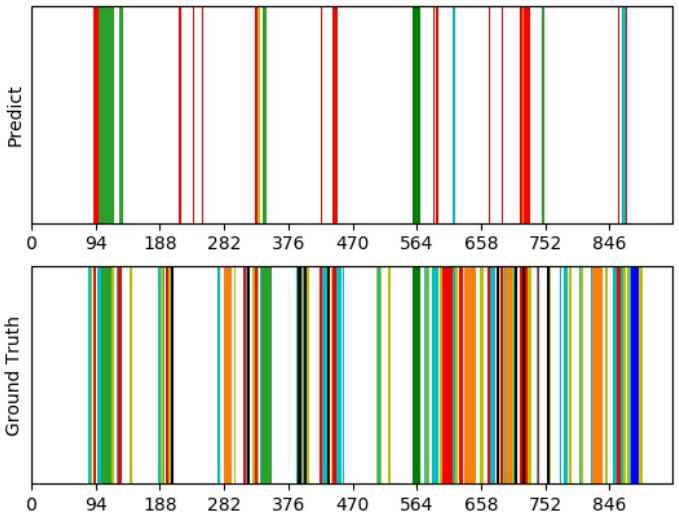
   由上述 accuracy 來看，accuracy 在每部影片中都大致上有 0.5 左右。以下實驗取第 1,2 部作圖，觀察其差異
   Labels: 分別對應到給予的 label（0-10）



OP01-R03-BaconAndEggs



OP02-R04-ContinentalBreakfast



可看到其實兩者產生很多白色的部分，也就是 label 0 (Other 的部分)，可能是因爲 ground truth 其實就有很多 unknown，導致 label 0 的 bias 很重，使得 predict 出來的結果很容易產生 0。另一個可能就是 random sample 所導致的結果，由於 training data 的 label 0 太多，加上 random sample,可能一些重要的時間點 model 其實並沒有學到,反而 input 的時候學到很多 0，導致這樣的結果。不過，值得注意的是，在第一部影片裡面有很大一部份是藍色的（label 7: Close），而 ground truth 同時也有一大部分是藍色的。這樣應該是在 random sample 的時候較容易 sample 到這一個區塊，讓 model 每次 training 都有看到這部分，因此可以產生對應的結果。