Report (HW1 - Classification)

學號： 311551149， 姓名： 朱彥勳

1. How to reproduce your result. (Including package, environment, reproduced method) (4 points)

   我的程式都是在 ubuntu 上面執行，只要下載所有 train.py 中 import 到的套件，輸入指令 python3 train.py，就可以執行 training 和 validation 的步驟，並再結束後輸出 train 和 validation 隨著 epoch 數對應到的 loss 作圖，同時也將 validation accuracy 最高的 model weight 存入 HW1_311551149.pt。接著輸入指令 python3 test.py 執行 test 的步驟。test.py 將 load 進剛剛 train.py 存入的 weight，同樣將資料夾 test 內的 image 送進組建好的 model，並同時將每一個 image 對應到的 image name 還有其 label 存進 csv 檔。

   一些 train.py 和 test.py 用到的 function 和 method，我連同遇到的困難寫在第五點”Problems encountered and discussion” 中。

2. Number of Model parameters (4 points): 8770

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 8, 224, 224]             224
       BatchNorm2d-2          [-1, 8, 224, 224]              16
              ReLU-3          [-1, 8, 224, 224]               0
         MaxPool2d-4          [-1, 8, 112, 112]               0
            Conv2d-5          [-1, 8, 112, 112]             584
       BatchNorm2d-6          [-1, 8, 112, 112]              16
              ReLU-7          [-1, 8, 112, 112]               0
         MaxPool2d-8            [-1, 8, 56, 56]               0
            Conv2d-9            [-1, 8, 56, 56]             584
      BatchNorm2d-10            [-1, 8, 56, 56]              16
             ReLU-11            [-1, 8, 56, 56]               0
        MaxPool2d-12            [-1, 8, 28, 28]               0
           Conv2d-13            [-1, 8, 28, 28]             584
      BatchNorm2d-14            [-1, 8, 28, 28]              16
             ReLU-15            [-1, 8, 28, 28]               0
        MaxPool2d-16             [-1, 8, 7, 7]               0
           Linear-17                   [-1, 16]           6,288
             ReLU-18                   [-1, 16]               0
           Linear-19                   [-1, 16]             272
             ReLU-20                   [-1, 16]               0
           Linear-21                   [-1, 10]             170
================================================================
Total params: 8,770
Trainable params: 8,770
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 13.21
Params size (MB): 0.03
Estimated Total Size (MB): 13.82
----------------------------------------------------------------
```

3. Explain model structure (as detail as possible)(4 points)

帶入 model 之前，我先設定了 training parameters，其中包含 300 個 epochs（超過 200 會開始慢慢收斂）、0.0001 的 learning rate（增加或縮小 10 倍都只會創造較低的 accuracy）；並且我的 loss function 用的是 cross entropy；optimizer 用的是 Adam。

架構:

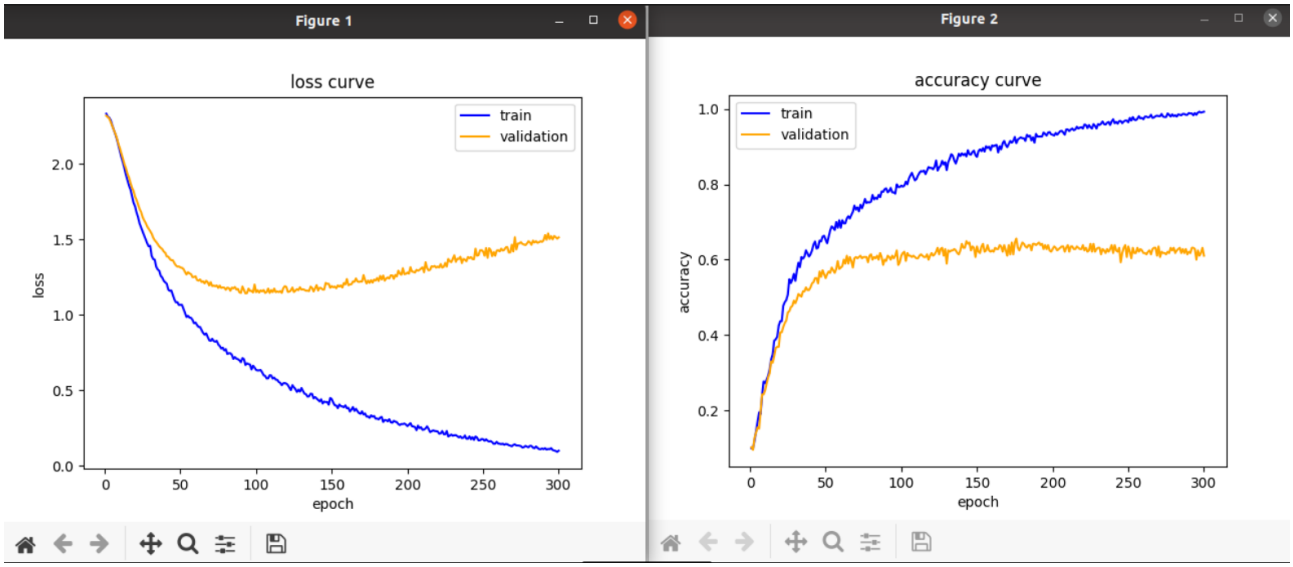
```
Network(
  (cnn_layers): Sequential(
    (0): Conv2d(3, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(8, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(8, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU()
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (12): Conv2d(8, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (14): ReLU()
    (15): MaxPool2d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  )
  (fc_layers): Sequential(
    (0): Linear(in_features=392, out_features=16, bias=True)
    (1): ReLU()
    (2): Linear(in_features=16, out_features=16, bias=True)
    (3): ReLU()
    (4): Linear(in_features=16, out_features=10, bias=True)
  )
)
```

架構的設計是在 fully connected layer 之前的四個 convolution layer 都搭配 batch normalization、ReLU 和 max pooling。每一層 conv.輸出都是 8，再小會嚴重影響 validation 的 accuracy；最後一個 max pooling 的 kernel size=4、stride=4，這樣的設計可以急速降低需要的 parameters 數量（將進原先的 1/4）。

第一層 fully connected layer 中輸出 16，也是一樣再小會嚴重影響 validation 的 accuracy，後面利用 ReLU 連接第二層 fully connected layer。第二層 fully connected layer 也是輸出 16，後面利用 ReLU 連接第三層 fully connected layer。第三層 fully connected layer 輸出 10，即為 class 的個數。

最後我的 parameters 總數是 8770。在程式的撰寫上，我用 cnn_layers 和 fc_layers 來為兩類的 layers 做區別。x = self.cnn_layers(x)可以直接把 cnn_layers 導進來，但在到 fc_layers 之前需要 flatten，可以想成是降維的動作，不然以（8, 7, 7）的形式不能直接進行 linear。最後用 x = self.fc_layers(x)就能把 x 給 return 出來。

4. Results (training and validation loss curve, training and validation accuracy curve) (4 points)



5. Problems encountered and discussion (4 points)

第一個問題就是對於 PyTorch 的整體架構不夠熟悉。在看完助教給的 tutorial 以及上網爬文後，我先將 train.py 分成幾個架構來釐清這些架構代表的意義和相關觀念。

1. 自訂 dataset 類別+將 dataset 轉為 dataloader: 在 SportLoader 這個 class 中，
   - ➔ init__得到 mode 的類別（train, valid...)、sport 負責讀下整個 csv 檔、img_name 將 csv 檔 column 0 的所有 str 存在一個 list 中、label 負責 column 1 的 list、transform 可以幫 data 做 augmentation（但我沒做)。這裡的困難點是 img_name 和 label 的建立，而我最後是用 iloc 來把 csv 的兩個 columns 分別寫進兩個 lists 裡面。
   - ➔ __len__ return image 的長度（個數）
   - ➔ __getitem__負責讀 image，這裡的困難點是由於讀進來的 image 會是（224, 224, 3)，但我看 model 的 summary 只能接受（3, 224, 224)，因此利用 31 行的 transpose 改變各維的數值。比較要注意的是 getitem 一次會回傳一張 image 和 image 的 target。最後把 dataset 轉為 dataloader，才能把 data 組合成 batch，並且能夠啟用 multiprocessing。
2. training model: model 的架構在第三點”explain model structure”有詳細說明。
3. 一些開始 train 之前的參數:
   - ➔ get device: 檢查能不能用 gpu，可以的話就用，反之就用 cpu
   - ➔ training parameters: 300 個 epochs、0.0001 的 learning rate（增加或縮小 10 倍都只會創造較低的 accuracy)
   - ➔ loss function, optimizer: cross entropy, Adam
   - ➔ model_path: 要存 weight 的路徑

4. 開始 training 和 validation: 在每一次的 epoch 中，以 batch size 為單位讀 image 進來，大部分都是要做 train 的基本架構，但我這裡遇到的困難是不清楚如何取得那個擁有 highest probability 的 class。後來我用_, train_pred = torch.max(outputs, 1)，因為我們並不關心那個最高的機率是多少，所以用_；我們關心的是那個最高機率的 index 是多少，所以把這個值存進 train_pred。再來用 train_acc += (train_pred.cpu() == labels.cpu()).sum().item()就能得到該次 epoch 的 training accuracy，但這有點像是多做的，頂多用來觀察有沒有 overfitting 的現象。同時用 train_loss += batch_loss.item()算出 train 的 loss。

   validation 的架構和 training 大同小異，只是要注意把 gradient calculation 給 disable 掉（用 with torch.no_grad()），也不需要去計算 gradient 和用 optimizer 去 update model。只是每次 epoch 結束前都要最佳的 validation accuracy 的 weight 記錄下來，我用的是 torch.save(model.state_dict(), model_path, _use_new_zipfile_serialization = False)。後面把 zipfile 給 false 掉好像是因為現在的版本會把所有存的東西放進 zip file 裡面，我把它 false 掉後就不會這樣了。

再來是 test.py 的相關概念

1. 建立屬於 test 的 SportLoader: 這裡遇到的困難是 test 的 SportLoader 需要經過修改。test 中不需要回傳 image 的 label 值，我改成回傳 image 還有 image 的路徑。目的是在跑 model 的時候可以在算出預測 label 時馬上把 image path 還有 label 同時寫進 list 裡面。
2. import training model: test 裡面也需要 model 才能跑出預測的 label，直接把這個 model 從 train.py 貼過來或是用 import 的都可以。
3. 一些開始 test 之前的參數:
   ➔ get device: 檢查能不能用 gpu，可以的話就用，反之就用 cpu
   ➔ model_path: 現在是要 load weight 的路徑
4. 開始 test: 一樣先把把 gradient calculation 給 disable 掉（用 with torch.no_grad()），在一次讀進 64（batch size）的 image 數中，先把 image name（test_loader 的第二個 return 值）append 進 list，再把這張 image 的 predicted label append 進另一個 list。最後用 with open('HW1_311551149.csv', 'w') as f: 把兩個 list 的值一個一個 row 寫進去。