學號：R07942091 系級： 電信碩一 姓名：許博閔

1. (2%) 請說明你實作的 CNN model，其模型架構、訓練參數和準確率為何？並請用與上述 CNN 接近的參數量，實做簡單的 DNN model，同時也說明其模型架構、訓練參數和準確率為何？並說明你觀察到了什麼？

   答：CNN 模型架構 :下面是 print(model)的結果，共有 5 層的 convolutional layer 和 3 層的 fully connected layer，總共的 trainable parameters 為 : 5758791，normalization 和 data augmentation 都有做，詳細方法寫在第三題。

```
Net(
  (conv0): Sequential(
    (0): Conv2d(1, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): LeakyReLU(negative_slope=0.05)
    (2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout2d(p=0.2)
  )
  (conv1): Sequential(
    (0): Conv2d(64, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): LeakyReLU(negative_slope=0.05)
    (2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Dropout2d(p=0.25)
  )
  (conv2): Sequential(
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): LeakyReLU(negative_slope=0.05)
    (2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Dropout2d(p=0.3)
  )
  (conv3): Sequential(
    (0): Conv2d(128, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): LeakyReLU(negative_slope=0.05)
    (2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Dropout2d(p=0.4)
  )


  (conv4): Sequential(
    (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): LeakyReLU(negative_slope=0.05)
    (2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Dropout2d(p=0.5)
  )
  (fc1): Sequential(
    (0): Linear(in_features=4608, out_features=512, bias=True)
    (1): ReLU()
    (2): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.5)
  )
  (fc2): Sequential(
    (0): Linear(in_features=512, out_features=512, bias=True)
    (1): ReLU()
    (2): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.5)
  )
  (fc3): Linear(in_features=512, out_features=7, bias=True)
)
```

   訓練參數：
   Conv2D : 前 2 層的 filter size = 5，後 3 層的 filter size = 3，stride 都 = 1，padding 的部分則是讓圖的大小經過 filter 後不變，除了第一層，都有 Maxpooling(size=2)

Activation function : CNN 的部分通過 LeakyRELU (斜率 = 0.05)，FC 的部分則是經過 RELU
BatchNormalization : 每層都有用，皆用 pytorch 預設參數
Dropout : dropout rate 依序為 0.2，0.25，0.3，0.4，0.5，0.5，0.5
Optimizer : Adam ， learning rate = 0.001
Loss function : nn.CrossEntropyLoss()

準確率 : kaggle public : 0.71914， kaggle private : 0.70660

答 ：以接近的參數量訓練出的 DNN model : 下面是 print(model)的結果，有 3 層的 fully connected layer，總共的 trainable parameters 為 : 5,778,439，比 CNN 多了約 2 萬個參數，normalization 和 data augmentation 的方法都和 CNN 一樣。

```
Net(
  (fc1): Sequential(
    (0): Linear(in_features=2304, out_features=2048, bias=True)
    (1): ReLU()
    (2): BatchNorm1d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.4)
  )
  (fc2): Sequential(
    (0): Linear(in_features=2048, out_features=512, bias=True)
    (1): ReLU()
    (2): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.5)
  )
  (fc3): Linear(in_features=512, out_features=7, bias=True)
)
```

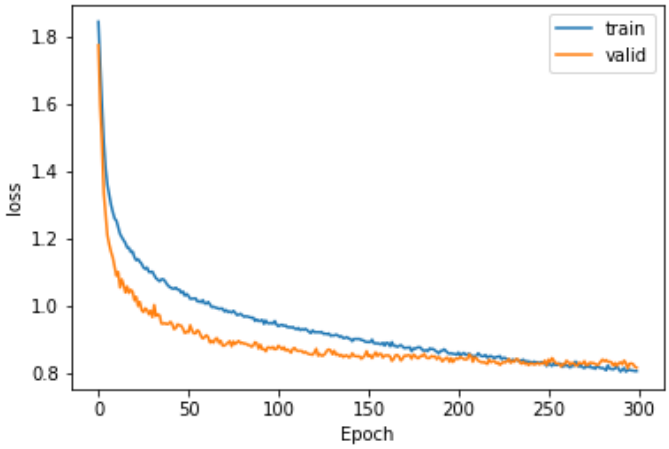
訓練參數 :
Activation function : FC 皆通過 RELU
BatchNormalization : 每層都有用，皆用 pytorch 預設參數
Dropout : dropout rate 依序為 0.4、0.5
Optimizer : Adam ， learning rate = 0.001
Loss function : nn.CrossEntropyLoss()

準確率 : train accuracy : 0.42~0.43 ， valid accuracy : 0.425~0.435

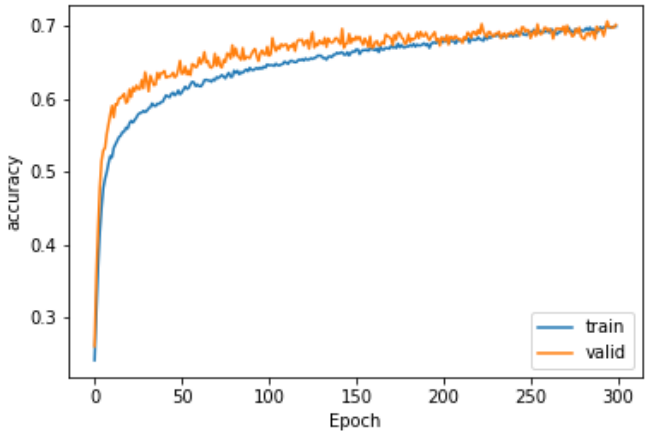
觀察 :
(1)在相同 epoch 的狀況下，CNN model 的準確率比 DNN 好很多。
(2)DNN 的訓練速度較快，應該是因為 CNN 需要的運算量較大。
(3)DNN 因為少了 CNN 提出圖片的特徵，似乎無法有效處理經過 data augmentation 後的圖片，因此準確率很低。

2. (1%) 承上題，請分別畫出這兩個 model 的訓練過程 (i.e., loss/accuracy v.s. epoch)
(Collaborators:)

答：CNN model 訓練過程 : 圖一 : loss v.s. epoch，圖二 : accuracy v.s. epoch，大約在第 250epoch 後開始 overfitting，最終 valid accuracy 在 0.69~0.70 間波動

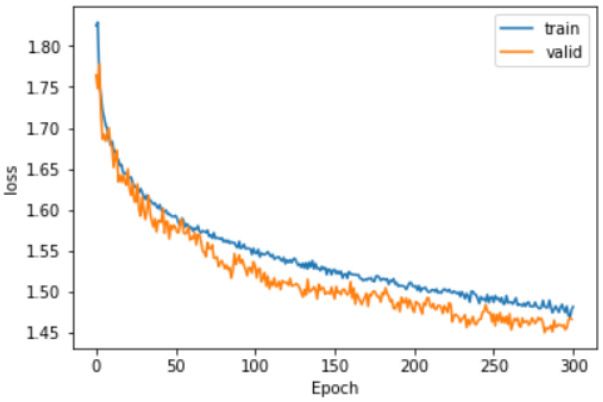


圖一

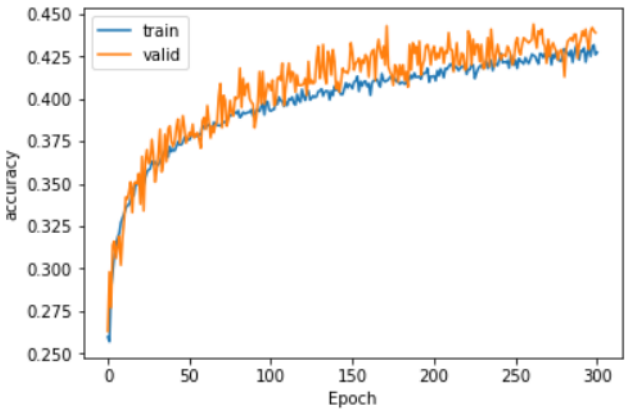


圖二

DNN model 訓練過程 : 圖三 : loss v.s. epoch，圖四 : accuracy v.s. epoch，可以看出 CNN model 表現好很多



圖三



圖四

3. (1%) 請嘗試 data normalization, data augmentation,說明實作方法並且說明實行前後對準確率有什麼樣的影響？

答：data normalization 和 data augmentation 都是用 torchvision 內的 transforms

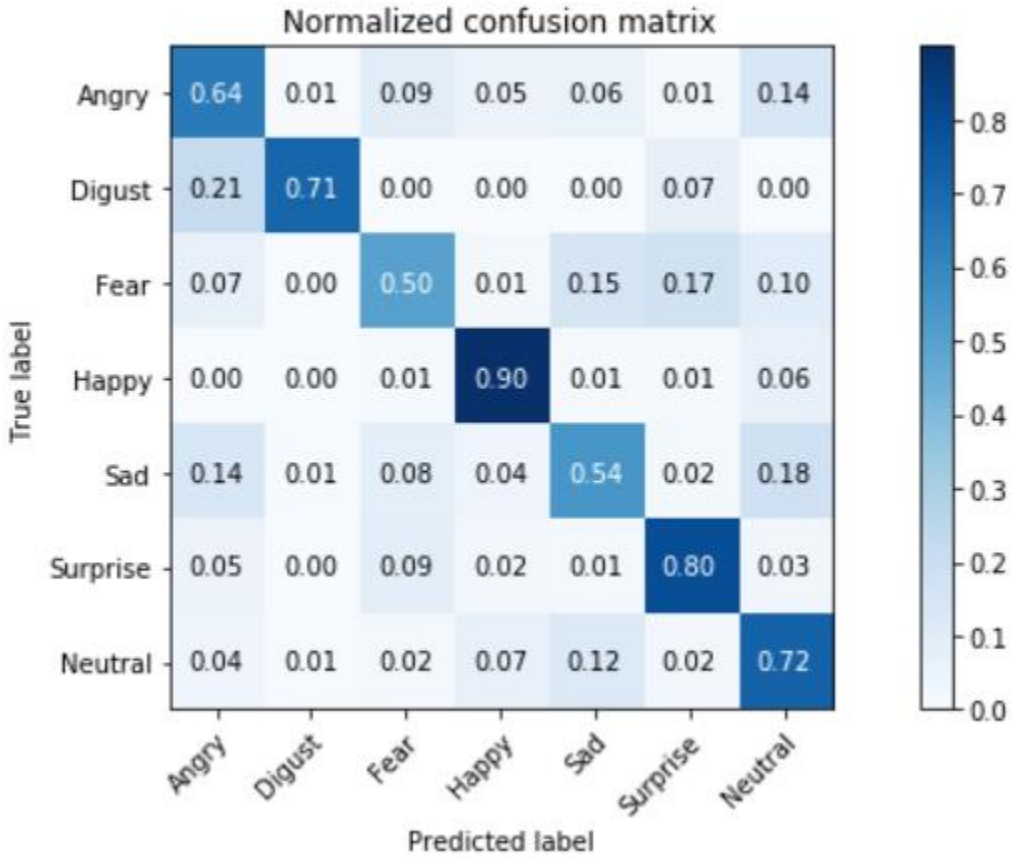
data normalization : transforms.ToTensor() 會把 pixel 值都 / 255，再用 transforms.Normalize([mean], [std], inplace=False) 做 normalization，mean 和 std 都是先算好的。做 normalization 可以使訓練的時候 loss 下降的速度比較快，並得到較好的準確率。

data augmentation : 用 transforms.RandomAffine(0, translate=(0.1,0.1), scale=(0.9,1.1), shear=10, fillcolor=0) 來達到圖片的平移、縮放和推移，旋轉和翻轉的部分則是用 transforms.RandomRotation()和 transforms.RandomHorizontalFlip()來達成。實行後能有效的避免 model overfitting，在 valid accuracy 上得到更高的準確率，但如果這些 function 的數值調太大，會導致 training 的 loss 無法下降。

	Loss 下降速度	準確率(validation data)
都沒有	次之	最低
Only normalization	最快	次之
Normalization+augmentation	最慢	最高

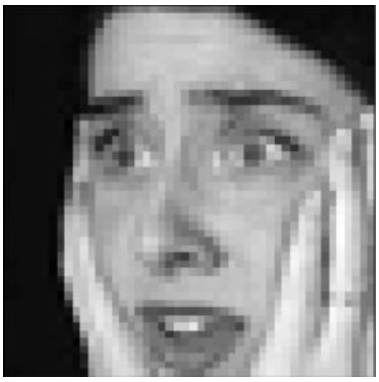
4. (1%) 觀察答錯的圖片中，哪些 class 彼此間容易用混？[繪出 confusion matrix 分析]

答：下圖是 validition data 畫出的 confusion matrix，可以看出我的 model 在分辨 Fear 和 Sad 這兩種情緒的正確率最低，因此從 validation image 中挑這兩類且答錯的圖片來觀察。除此之外，Fear/Surprise、Sad/Neutral 之間也容易用混。



下面 2 張圖就是 model 預測錯誤的圖，以我的角度來看，我認為這兩張圖不論是被歸類為 Fear 或 Sad 應該都不算答錯，而且兩張圖的動作也很類似，都把雙手放到臉頰上。因此這種答案看似模稜兩可的圖片，model 較容易弄混。

True:Fear，Prediction:Sad



True:Sad，Prediction:Fear