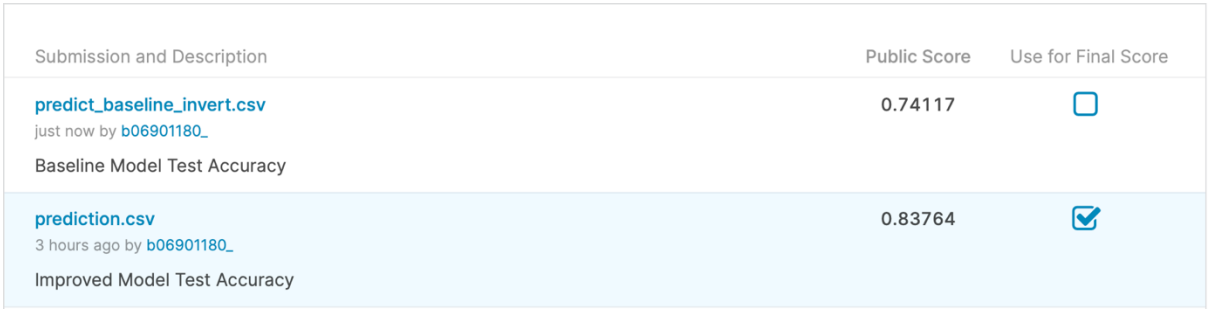
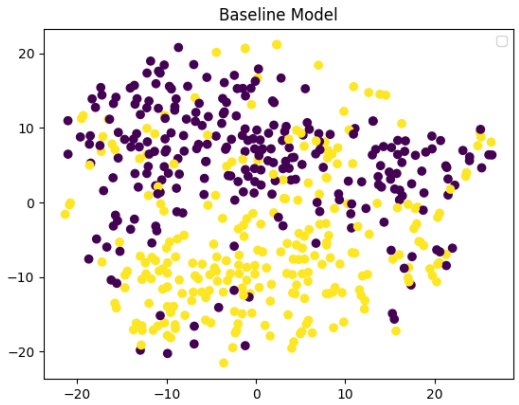
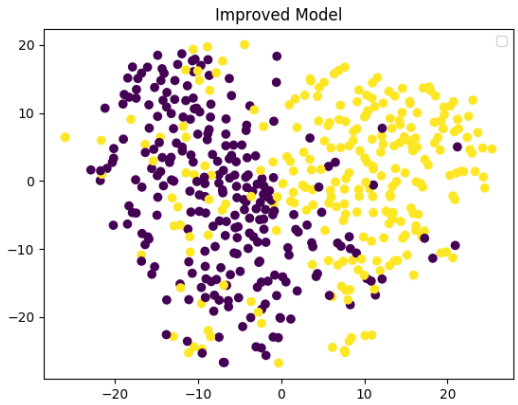
學號：B06901180　系級：電機三　　姓名：鄭謹譯

1. (3%) 請至少使用兩種方法 (autoencoder 架構、optimizer、data preprocessing、後續降維方法、clustering 算法等等) 來改進 baseline code 的 accuracy。
   a. 分別記錄改進前、後的 test accuracy 為多少。
   b. 分別使用改進前、後的方法，將 **val data** 的降維結果 (embedding) 與他們對應的 label 畫出來。
   c. 盡量詳細說明你做了哪些改進。







On validation data:

Accuracy of Baseline Model 55%　Accuracy of Improved Model 82%

改進方法：改變 autoencoder 架構 （ Encoder ）(成功)

1.在每層 Conv2d、ReLU 中間加入 BatchNorm2d（參考作業三）

2.多加 Conv2d(256,512,3,1,1)、BatchNorm2d(512)、ReLU、MaxPool2d ，將最後的 Channel 數目增加到 512，MaxPool2d 多一層史最小為 2*2，理想能從中取得更佳資訊，濾掉 Noise

改進方法：改變 autoencoder 架構 （ Decoder) (成功)

配合 Encoder 增加，必須調整 Decoder 參數，嘗試過以下方式

ConvTranspose2d(512,256,4) ReLU() ConvTranspose2d(256,128,7) ReLU() ConvTranspose2d(128,64,9) ReLU() ConvTranspose2d(64,3,14) ReLU()	ConvTranspose2d(512,128,7) ReLU() ConvTranspose2d(128,64,9) ReLU() ConvTranspose2d(64,3,17) ReLU()
Kaggle Score : NA (val acc is far low than others)	Kaggle Score : 0.80729
ConvTranspose2d(512,181,7) ReLU() ConvTranspose2d(181,64,9) ReLU() ConvTranspose2d(64,3,17) ReLU()	ConvTranspose2d(512,181,7) ReLU() ConvTranspose2d(181,64,11) ReLU() ConvTranspose2d(64,3,15) ReLU()
Kaggle Score : 0.82070	Kaggle Score : 0.83764

NA: No submission (val acc is far low than others)

1.測試結果 4 層 Decoder 結果遠差於其他 3 層的 Decoder，val acc 0.67，3 層的皆在 0.75 以上

2. 3 層中又以 512 181 64 3 的結構為佳，181=2^7.5

最終 AE 架構如下

```
AE(
  (encoder): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU(inplace=True)
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU(inplace=True)
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (12): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (14): ReLU(inplace=True)
    (15): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (decoder): Sequential(
    (0): ConvTranspose2d(512, 181, kernel_size=(7, 7), stride=(1, 1))
    (1): ReLU(inplace=True)
    (2): ConvTranspose2d(181, 64, kernel_size=(11, 11), stride=(1, 1))
    (3): ReLU(inplace=True)
    (4): ConvTranspose2d(64, 3, kernel_size=(15, 15), stride=(1, 1))
    (5): Tanh()
  )
)
```

改進方法：改變 Optimizer ---- 訓練一半後轉為 SGD (失敗)

使用上述 80.729 架構時，改變 Optimizer 結果為 79.976，下降 0.753%

改進方法：Data preprocessing ---- Normalization (失敗)

Normalize 後 Reconstruction 的色調跑掉

使用上述 80.729 架構時，Normalize 結果為 79.952，下降 0.777%

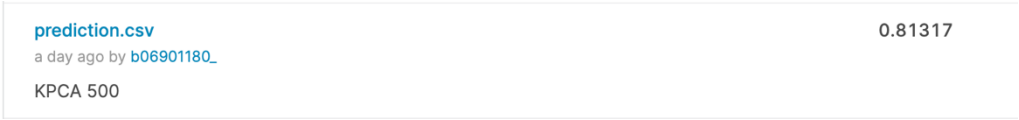
改進方法：Data preprocessing ---- Add Noise (失敗)

1. 加入 randn 的 noise，loss function 使用未加 noise 的圖片計算，val 為 73%，較未加入的 82%低許多
2. 加入 randn 的 noise，loss function 使用加入 noise 的圖片計算，val 為 79%，較未加入的 82%低但比上述方法高些

因皆變差不少，故未上傳 Kaggle，無 testing 分數

改進方法：後續降維方法 ---- KPCA n_components=200 -> 其他 (失敗)

嘗試過 1-500 間的許多數值，皆變差。僅上傳 500 的結果



故維持 n_components=200

改進方法：後續降維方法 ---- TSNE perplexity=30.0, early_exaggeration=12 -> 其他 (失敗)

亂數分別嘗試兩個變數（各 20 種）使用 val acc 來判斷，後來發現預設的 30,12 為最佳，故維持預設參數

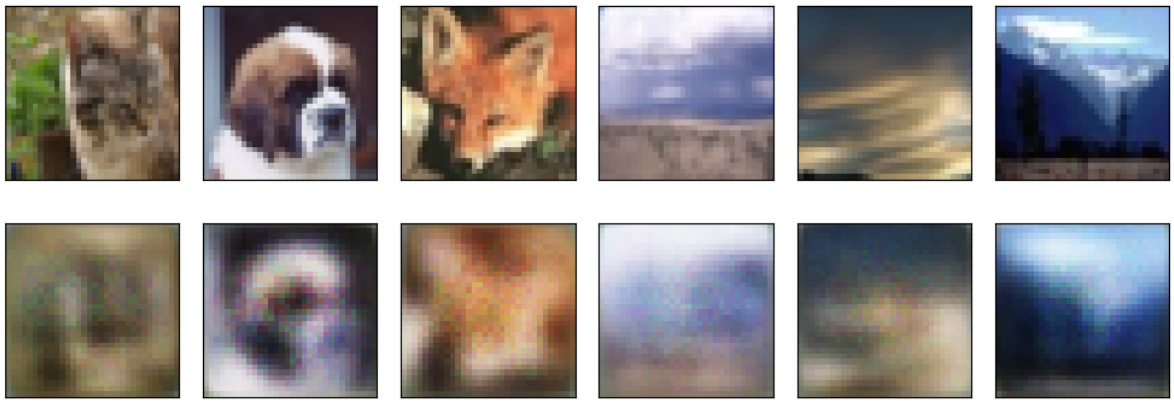
改進方法：clustering 算法 ---- MiniBatchKmeans vs Kmeans (無太大差異)

pred = MiniBatchKMeans(n_clusters=2, random_state=0).fit(X_embedded)

pred = KMeans(n_clusters=2, random_state=0).fit(X_embedded)

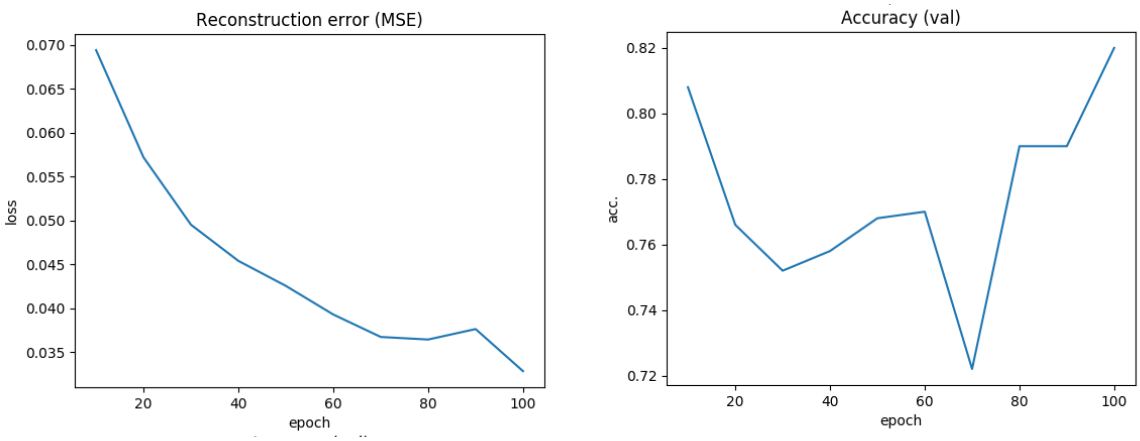
2. (1%) 使用你 test accuracy 最高的 autoencoder，從 trainX 中，取出 index 1, 2, 3, 6, 7, 9 這 6 張圖片

a. 畫出他們的原圖以及 reconstruct 之後的圖片。 Kaggle Score : 0.83764



上列為原圖，下列為 reconstruct

3. (2%) 在 autoencoder 的訓練過程中，至少挑選 10 個 checkpoints
   a. 請用 model 的 train reconstruction error (用所有的 trainX 計算 MSE) 和 **val accuracy** 對那些 checkpoints 作圖。
   b. 簡單說明你觀察到的現象。



MSE 大致上隨 Epoch 增加而減少，符合預期結果。

Val acc 則出乎意料，一開始 10 個 epoch 接近 0.81，之後下降一直到 epoch=100 截止訓練時才超過 epoch = 10 的 acc，為 0.82。

另外助教給的參考程式碼 sort 時會按照檔名讀入，然而順序為_10 _100 _20 _30 以此類推，故需要另外調整 epoch = 100 的位置。因此也可以解釋 Colab 上為何第二個點的 error 最小，acc 最高，其實它為最後一個 checkpoint。（見右圖）

上述 improved model 的圖已修正此問題。