Machine Learning HW9 Report

學號：B06901045 系級：電機三 姓名：曹林熹

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。

ANS：

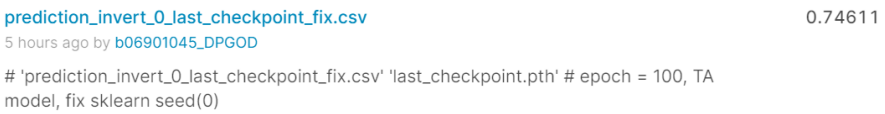
a. 分別記錄改進前、後的 test accuracy 為多少。

1) TA model:

這個部分我並沒有對任何的 model 做改變，基本上直接拿助教給的 sample code 跑過一次後直接丟到 kaggle 上面去預測。中間的 training 過程，助教很貼心的將 random seed 都已經設置好，然而在 predict 過程中，我在 sklearn 的三個套件含式庫都多加了 random state(0)，因為經過好幾次 debug 後，發現不加的話 prediction 會無法 reproduce，算是一個要注意的地方。

```
from sklearn.decomposition import KernelPCA
from sklearn.manifold import TSNE
from sklearn.cluster import MiniBatchKMeans
```

結果發現， sample code 可以輕鬆地過 simple baseline，因此我嘗試用不同的方法去突破 strong baseline。



2) change the autoencoder model:

這部分我只採取使用優化的 autoencoder 去達成我的 strong baseline，其他的 optimizer、data preprocessing、後續降維方法、clustering 算法等等都為 sample code。我觀察了助教的 autoencoder，可以給出結構如下：

```
[10] model = AE().cuda()

     from torchsummary import summary
     summary(model, input_size=(3, 32, 32))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 32, 32]           1,792
              ReLU-2          [-1, 64, 32, 32]               0
         MaxPool2d-3          [-1, 64, 16, 16]               0
            Conv2d-4         [-1, 128, 16, 16]          73,856
              ReLU-5         [-1, 128, 16, 16]               0
         MaxPool2d-6           [-1, 128, 8, 8]               0
            Conv2d-7           [-1, 256, 8, 8]         295,168
              ReLU-8           [-1, 256, 8, 8]               0
         MaxPool2d-9           [-1, 256, 4, 4]               0
  ConvTranspose2d-10           [-1, 128, 8, 8]         819,328
             ReLU-11           [-1, 128, 8, 8]               0
  ConvTranspose2d-12          [-1, 64, 16, 16]         663,616
             ReLU-13          [-1, 64, 16, 16]               0
  ConvTranspose2d-14           [-1, 3, 32, 32]          55,491
             Tanh-15           [-1, 3, 32, 32]               0
================================================================
Total params: 1,909,251
Trainable params: 1,909,251
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 2.39
Params size (MB): 7.28
Estimated Total Size (MB): 9.69
----------------------------------------------------------------
```

因為我們有限制 training 的上限為 30 min，我每次 training 都是使用 epoch = 100，平均下來每次 epoch 不可超過 18(sec)。觀察助教的 model parameters，我嘗試自己多加了一層 cnn layer，並且微調了 decoder，結構如下(詳細說明在 1(c))：

```
[11] model = AE().cuda()

     from torchsummary import summary
     summary(model, input_size=(3, 32, 32))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 32, 32]           4,864
       BatchNorm2d-2          [-1, 64, 32, 32]             128
              ReLU-3          [-1, 64, 32, 32]               0
         MaxPool2d-4          [-1, 64, 16, 16]               0
            Conv2d-5         [-1, 128, 16, 16]          73,856
       BatchNorm2d-6         [-1, 128, 16, 16]             256
              ReLU-7         [-1, 128, 16, 16]               0
         MaxPool2d-8           [-1, 128, 8, 8]               0
            Conv2d-9           [-1, 256, 8, 8]         295,168
      BatchNorm2d-10           [-1, 256, 8, 8]             512
             ReLU-11           [-1, 256, 8, 8]               0
        MaxPool2d-12           [-1, 256, 4, 4]               0
           Conv2d-13           [-1, 512, 4, 4]       1,180,160
      BatchNorm2d-14           [-1, 512, 4, 4]           1,024
             ReLU-15           [-1, 512, 4, 4]               0
        MaxPool2d-16           [-1, 512, 2, 2]               0
  ConvTranspose2d-17           [-1, 128, 8, 8]       3,211,392
             ReLU-18           [-1, 128, 8, 8]               0
  ConvTranspose2d-19          [-1, 64, 16, 16]         663,616
             ReLU-20          [-1, 64, 16, 16]               0
  ConvTranspose2d-21           [-1, 3, 32, 32]          55,491
             Tanh-22           [-1, 3, 32, 32]               0
================================================================
Total params: 5,486,467
Trainable params: 5,486,467
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 3.47
Params size (MB): 20.93
Estimated Total Size (MB): 24.41
----------------------------------------------------------------
```

最後使用此 model 做預測，發現 kaggle score 非常的好，突破了 strong

baseline，並且也可以順利 reproduce。

prediction_1_checkpoint_100_fix.csv 0.78470
5 hours ago by b06901045_DPGOD
'prediction_1_checkpoint_100_fix' 'checkpoint_100.pth' # epoch = 100, encode = 4, decode = 3, para = 5,486,467, fix sklearn seed(0)

3) change the autoencoder model and use Kmeans cluster:
為了運用兩種方法增強我的 baseline model，上面使用第一種方法為 change autoencoder，接下來我運用 Kmeans 取代 MiniBatchKmeans，同樣使用上面改過的 autoencoder 作為架構。因為參考網路文獻，MiniBatchK-Means 為了避免樣本量太大時的計算難題，讓算法收斂速度大大加快的同時，造成的損失代價是聚類精確度下降。方法很簡單，import Kmeans 並且改成我標註的那行就好。

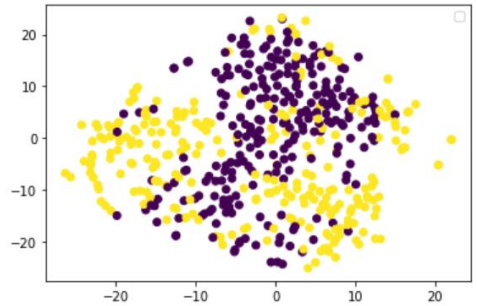
```
# Clustering
# pred = MiniBatchKMeans(n_clusters=2, random_state=0).fit(X_embedded) #使用 MiniBatchKMeans
pred = KMeans(n_clusters=2, random_state=0).fit(X_embedded) #使用 KMeans
```

丟到 kaggle 後，沒想到成績大幅提升。

prediction_1_checkpoint_100_kmeans_repro.csv 0.81294
21 minutes ago by b06901045_DPGOD
'prediction_1_checkpoint_100_kmeans_repro.csv' 'checkpoint_100.pth' # epoch = 100, encode = 4, decode = 3, para = 5,486,467, fix sklearn seed(0), cluster = kmeans # acc = # 使用 random_state 後可以 repro

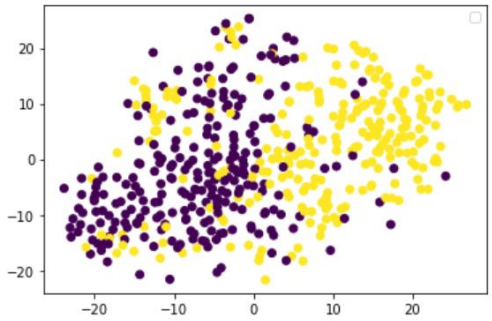
b. 分別使用改進前、後的方法，將 val data 的降維結果（embedding）與他們對應的 label 畫出來。
1) TA model:

```
Latents Shape: (500, 4096)
First Reduction Shape: (500, 200)
No handles with labels found to put in legend.
Second Reduction Shape: (500, 2)
The clustering accuracy is: 0.6719999999999999
The clustering result:
```



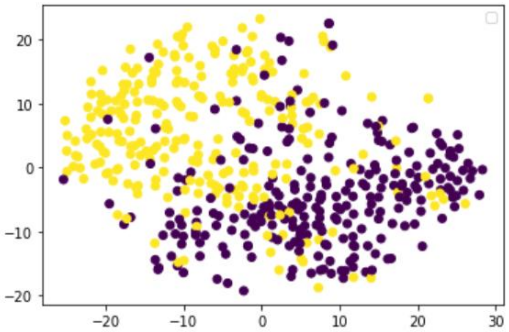
2) change the autoencoder model:

```
Latents Shape: (500, 2048)
First Reduction Shape: (500, 200)
No handles with labels found to put in legend.
Second Reduction Shape: (500, 2)
The clustering accuracy is: 0.746
The clustering result:
```



3) change the autoencoder model and use Kmeans cluster:

```
Latents Shape: (500, 2048)
First Reduction Shape: (500, 200)
No handles with labels found to put in legend.
Second Reduction Shape: (500, 2)
The clustering accuracy is: 0.778
The clustering result:
```



c. 盡量詳細說明你做了哪些改進。

1) change the autoencoder model:

前面在 a 小題已經有提到我們的 model 架構，主要是我又多 add 一層 cnn layer，加的方法是加在最後一層，並且經過 maxpooling 後，每個 feature 成為 2*2 pixel。經過 hw3 的練習，實作後認為把 cnn 層數多加後，會得到比較好的重點 feature，還原後自然也會成為比較好的圖片。不過一旦把層數多加，我們的參數會過多(做太多次卷積運算)，這樣會超過 30 min 的訓練限制，因此我 encoder 多加一層 cnn，encoder 稍微更動第一層的

nn.ConvTranspose2d(256, 128, 5, stride=1) =>

nn.ConvTranspose2d(512, 128, 7, stride=1)，

其餘不更動，總共成為 500 多萬參數的 model。Training 後發現我 epoch 每

輪只花 3~4 sec，符合訓練時間限制。

2) change the autoencoder model and use Kmeans cluster:
改成 Kmeans 方法，有在上面提過。

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

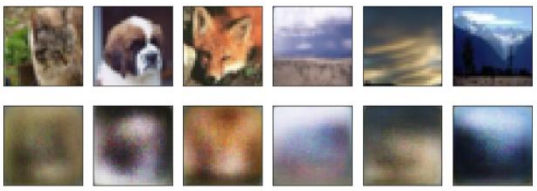
ANS：

a. 畫出他們的原圖以及 reconstruct 之後的圖片。

由於原圖的 pixel 量也不是很多 (32*32 RGB)，因此還原回來後的圖片本身就很難達到用肉眼馬上就知道那是什麼，但是經由觀察後，可以發現機器仍然有把重點輪廓給標示出來，只有少部分線條並沒有清楚的呈現(ex. 狐狸眼睛、風景的樹線條等)，總結而言仍是一個不錯的 reconstruct decoder。左圖為我最好的 autoencoder，右圖為助教的 sample code autoencoder，可以發現我優化過的 model 確實有明顯較好的 reconstruction。

Mine

TA



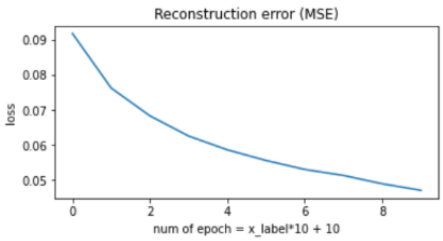
3. (2%) 在 autoencoder 的訓練過程中，至少挑選 10 個 checkpoints

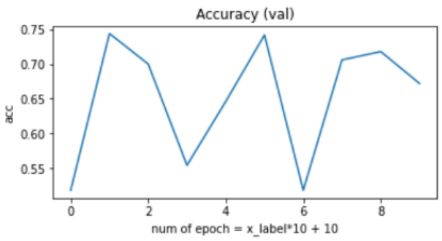
ANS：

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。

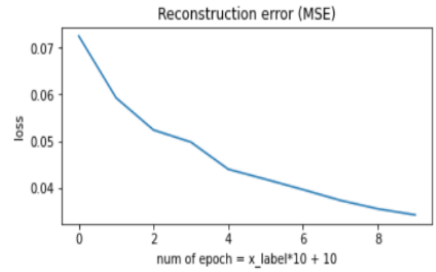
我每次經過 10 個 epoch 就會存我的 checkpoint，總共跑 100 次，因此 checkpoint_100.pth = last_checkpoint.pth。

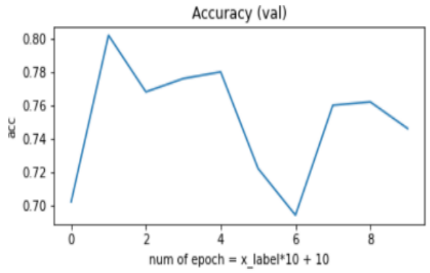
1) TA model:



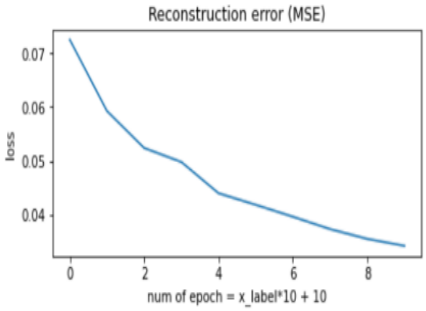


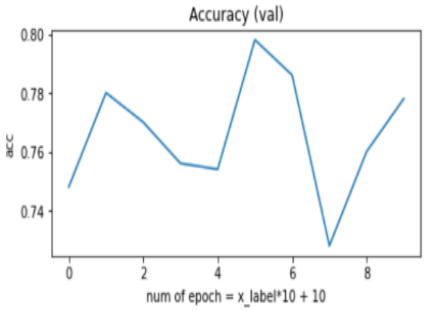
2) change the autoencoder model:





3) change the autoencoder model and use Kmeans cluster:





b. 簡單說明你觀察到的現象。

可以發現不管使用什麼方法，每次得到的 reconstruction loss 會因為 epoch 增加而下降，然而 accuracy 就不一定會因此上升。可以看到 TA model 反而在 epoch = 20 or 60 時有最高的 acc。在我的 change the autoencoder model 中，可以看到在 epoch = 20 時也有最高的 acc，使用最終的 model checkpoint 不一定會在 val_set 表現特別好的預測準確度。同樣地，在我的 change the autoencoder model and use Kmeans cluster 中，當 epoch = 60 看似有最高的準確率，但最終我們使用 epoch = 100，其實效果也是不差。