Machine Learning HW10 Report

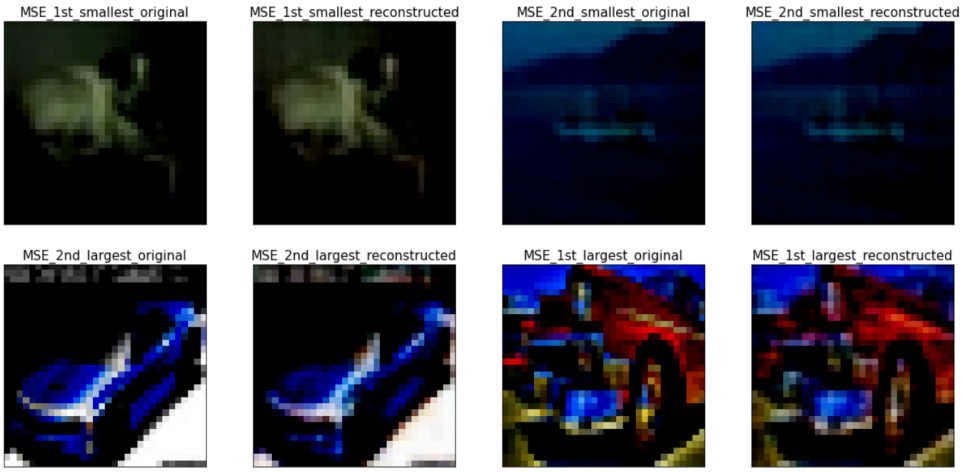
學號：B06901045 系級：電機三 姓名：曹林熹

1. **(2%) 任取一個 baseline model (sample code 裡定義的 fcn，cnn，vae) 與你在 kaggle leaderboard 上表現最好的 model (如果表現最好的 model 就是 sample code 裡定義的 model 的話就再任選一個，e.g. 如果 cnn 最好那就再選 fcn)，對各自重建的 testing data 的 image 中選出與原圖 mse 最大的兩張加上最小的兩張並畫出來。(假設有五張圖，每張圖經由 autoencoder A 重建的圖片與原圖的 MSE 分別為 [25.4, 33.6, 15, 39, 54.8]，則 MSE 最大的兩張是圖 4、5 而最小的是圖 1、3)。須同時附上原圖與經 autoencoder 重建的圖片。(圖片總數：(原圖+重建)*(兩顆 model)*(mse 最大兩張+mse 最小兩張) = 16 張)**

   ANS:

   a. baseline model：

      這裡的 baseline model 我使用的是助教 sample code 中的 cnn，架構的部分就不重新贅述。可以看到我們畫出 MSE 最大、次大、最小、次小的原圖與還原圖，總共有八張圖。仔細觀察發現 MSE 比較小的圖片是原圖本身就很統一色系，所以還原回來好像都差不多，看不出差別。而我們觀察 MSE 較大的圖片可以發現色彩多為鮮艷的 (MSE 最大那張看起來是五彩繽紛的車)，但是用肉眼看好像也沒有特別的差異，不過或許對於機器而言總是能找出我們無法認知的差別，我猜想使用 best model 能夠更有效看出 anomaly detection。



      我們一樣也丟到 kaggle 看看成績，結果為 0.55259，可見仍有很大的進步空間。

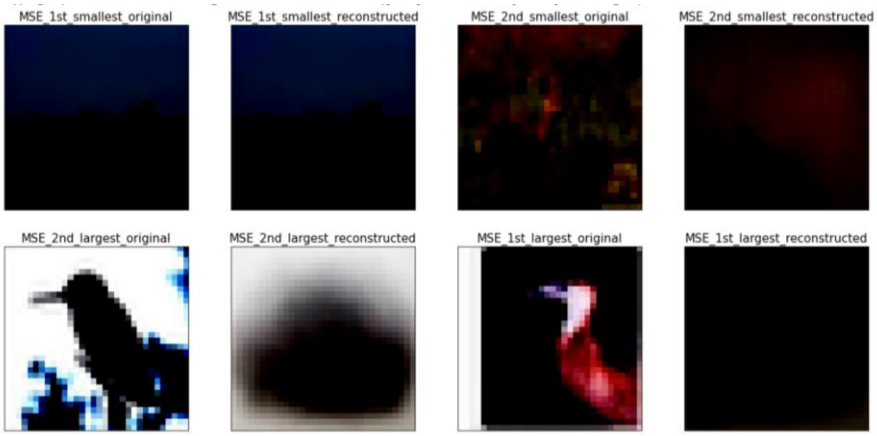
prediction_0.csv 0.55259
15 days ago by b06901045_DPGOD
TA

b. best model：

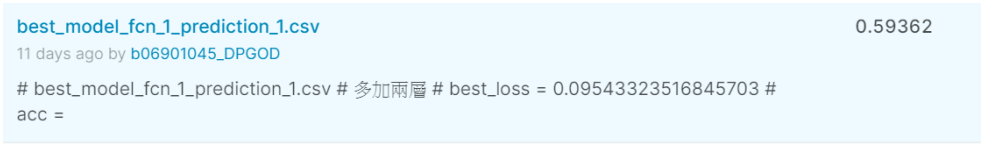
這裡的 best model 我使用的是我自己新架構的 fcn，架構的部分是我與 sample code fcn 比較上在 encoder 與 decoder 都各多加了兩層 linear layer，因此總共有 12 層，此目的是讓每一層的結構不要跳動太快（ex. 原本第一層是 nn.Linear(32 * 32 * 3, 128)，後來改成 nn.Linear(32 * 32 * 3, 512)，讓 input node 與 output node 不要落差太大）。

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                [-1, 1, 512]       1,573,376
              ReLU-2                [-1, 1, 512]               0
            Linear-3                [-1, 1, 256]         131,328
              ReLU-4                [-1, 1, 256]               0
            Linear-5                [-1, 1, 128]          32,896
              ReLU-6                [-1, 1, 128]               0
            Linear-7                 [-1, 1, 64]           8,256
              ReLU-8                 [-1, 1, 64]               0
            Linear-9                 [-1, 1, 12]             780
             ReLU-10                 [-1, 1, 12]               0
           Linear-11                  [-1, 1, 3]              39
           Linear-12                 [-1, 1, 12]              48
             ReLU-13                 [-1, 1, 12]               0
           Linear-14                 [-1, 1, 64]             832
             ReLU-15                 [-1, 1, 64]               0
           Linear-16                [-1, 1, 128]           8,320
             ReLU-17                [-1, 1, 128]               0
           Linear-18                [-1, 1, 256]          33,024
             ReLU-19                [-1, 1, 256]               0
           Linear-20                [-1, 1, 512]         131,584
             ReLU-21                [-1, 1, 512]               0
           Linear-22               [-1, 1, 3072]       1,575,936
             Tanh-23               [-1, 1, 3072]               0
================================================================
Total params: 3,496,419
Trainable params: 3,496,419
Non-trainable params: 0
```

預測好後可以看到我們畫出 MSE 最大、次大、最小、次小的原圖與還原圖，總共有八張圖。仔細觀察發現 MSE 比較小的圖片是原圖本身就暗暗的，還原看起來也都暗暗的。而我們觀察 MSE 較大的圖片可以發現 reconstructed 後的圖與原圖完全不同，有點像是雜訊之類的圖，這樣代表我們是有成功找出 noise，算是成功的 anomaly detection。



我們一樣也丟到 kaggle 看看成績，結果為 0.59362，比起 baseline model 有稍微進步 5%，但是距離過 strong baseline 仍有很大的挑戰。



2. **(1%) 嘗試把 sample code 中的 K-means 與 PCA 分別做在 autoencoder 的 encoder output 上，並回報兩者的 auc score 以及本來 model 的 auc。autoencoder 不限。不論分數與本來的 model 相比有上升還是下降，請同學簡述原因。**

ANS:

這邊我使用 fcn 作為我的 autoencoder 來看 kmeans 與 PCA 的效果，fcn 架構為助教 sample code，而不使用降維在 kaggle 上的成績為 0.58737。

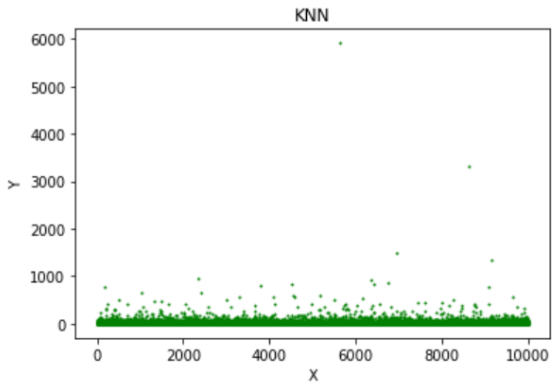
如何使用 encoder output 做降維，可以分為以下步驟。

a. get encode vector

這步驟是為了要取得 encoder output，那一樣使用 testing 的 block，不過取 output 時改成 output = model.encoder(img)，不做 decoder 部分。

b. K-Means

使用 K-Means 前我們要先將取好的 latent 變成合適的維度，經過處理後我們變成 train_latent = (40000, 3) 與 test_latent = (10000, 3) 的維度。總共分成 9 類，可以看到我們 10000 筆資料的分布狀況。



最後丟到 kaggle 得到 auc score，只有 0.50369。

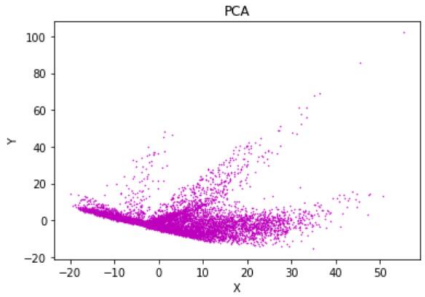
結果變低的原因我上網查了一下關於 K-Means 的缺點，可以分為四種：
(1) 對於離群點和孤立點敏感
(2) k 值選擇
(3) 初始聚類中心的選擇
(4) 只能發現球狀簇
以下我針對 (2)(3)(4) 來討論。
首先關於 (2) 的話，助教的 K-Means 是讓我們跑一個迴圈從 k = 0 to k = 9，最後我選取的是 k = 9 的預測結果，因為我使用 k = 5 時預測準確度只有 4 成左右，因此我認為要找到一個適合的 k 值是困難的事情，因為也沒有人能夠證明 k 愈大效果愈好。

其次，針對 (3) 的部分是參考網路文章提到有關於初始聚類中心的選擇優化，可以選擇批次距離盡可能遠的 K 個點。此部分受限於時間因此我並沒有重新實作，不過確實初始值對我們影響也很大。

最後，(4) 的部分是因為 K-Means 演算法所採取距離度量的方式，造成分群幾乎都是球狀分布，有時會無法顯示真實的資料分布。總結而言，K-Means 演算法在此情形下仍有許多考量的因素才可使用。

c. PCA

使用 PCA 前我們一樣要先將取好的 latent 變成合適的維度，經過處理後我們變成 train_latent = (40000, 3) 與 test_latent = (10000, 3) 的維度。可以看到降維部分變成很奇怪的形狀，但是又有點規則的感覺。



最後丟到 kaggle 得到 auc score，只有 0.50214，依樣是退步的情況。

同樣探討結果變低的原因我上網查了一下關於 PCA 的缺點，可以分為兩種：

(1) 主成分各個特徵維度不如原始樣本特徵，因此會造成大誤差

(2) 方差小的非主成分可能含有對樣本差異的重要資訊，降維丟棄此資訊會造成影響

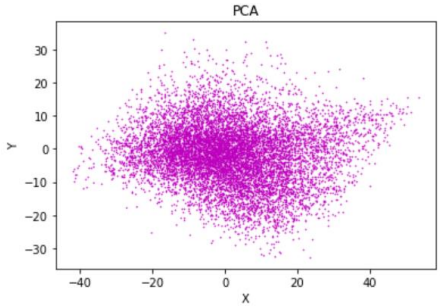
可以看出 PCA 的致命缺點就是降維，其實蠻合理的，因為使用降為勢必會將一些資訊給捨棄，因此上面兩個缺點造成準確度降低我認為是合理的情形。

3. **(1%) 如 hw9，使用 PCA 或 T-sne 將 testing data 投影在 2 維平面上，並將 testing data 經第 1 題的兩顆 model 的 encoder 降維後的 output 投影在 2 維平面上，觀察經 encoder 降維後是否分成兩群的情況更明顯。（因未給定 testing label，所以點不須著色）**

ANS:

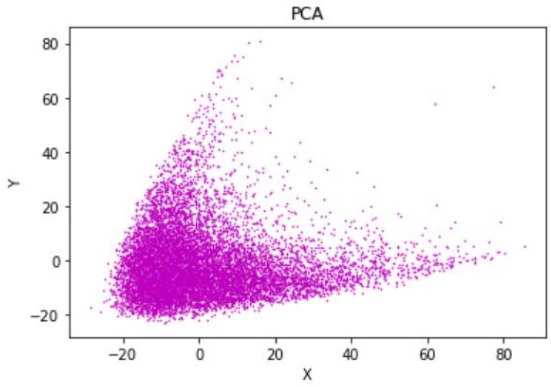
a. testing data without dimension reduction

沒有經過 model 的 PCA，可以看到每筆資料都很分散聚集在一起。



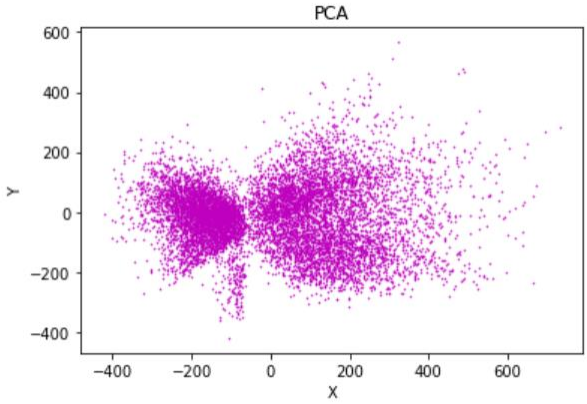
b. testing data with cnn baseline model

可以看到我們的圖片經過 PCA 降維後感覺上是有規律的形狀分布（類似在頂點時比較多聚集點的拋物線），這時感覺上有比較分出兩群，一群是靠近頂點的那一團，而另外一群則是靠近尾端的部分。



c. testing data with fcn best model

可以看到我們的圖片經過 PCA 降維後中間有一條白色的線分隔兩群，形成一個蝴蝶結的圖案，比起 baseline model，因為有明顯分隔線因此感覺上比較有分出兩群。



4. (2%) 說明為何使用 auc score 來衡量而非 binary classification 常用的 f1 score。如果使用 f1 score 會有什麼不便之處？

ANS:

這裡的分析我分為三個部分，第一部分為先備知識，第二部分探討 f1 score，的三部分則是探討 auc socre。

a. 第一部分 先備知識）

要了解 f1 score 與 auc score，首先我們要了解最基本來判定模型好壞的評斷方法，最廣為人知的就是 ACC 了，主要是拿正確預測資料來除以所有資料，不過會產生一個問題就是如果資料分布多為偏向一種特定類別的話，那機器容易判定所有資料為此類別，雖然準確率高，但是是因為某一特定類別的資料數本來就大造成的結果。

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{P + N}$$

為了避免此狀況發生，有了 Recall 與 Precision 的評斷標準產生，Precision 看的是在預測正向的情形下，實際的「精準度」是多少，而 Recall 則是看在實際情形為正向的狀況下，預測「能召回多少」實際正向的答案。同時，也可看到 Precision 和 Recall 都不考慮 True Negative，因為通常 True Negative 會是答對的 Null Hypothesis，也就是我們不感興趣的正確結果。採用此作法可以將資料分布不均造成的模型預測情形有所改善，不過仍然不足的是，判斷依據到底是要採用 Recall 還是 Precision，接著 f1 score 出現了。

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{P}$$

$$Precision = \frac{TP}{TP + FP}$$

b. 第二部分 f1 score）

概念很簡單，由以下公式將 Recall 與 Precision 都考慮進去，成為了當今很廣泛使用的評斷標準。不過儘管這個公式很常被大家使用，但是仍有缺點。舉例來說，當我的兩個模型一個 Precision 特別高、recall 特別低，但是另一個 Recall 特別高、Precision 特別低，此時 f1 score 可能是差不多的，但我們也無法依據此衡量到底要用哪個模型較好。

$$F1 = \frac{2TP}{2TP + FN + FP} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

c. 第三部分 auc score）

此次作業使用這種方法，評斷方式是採用 ROC 曲線的面積。ROC 曲線是針對每個 threshold 去標點（ex.超過 0.5 就算那個類別，0.5 就是我們的 threshold），把這些點連起來形成曲線，因此可以把看到不同 threshold 的趨勢圖。採用 auc 來算面積的話，我們可以得知”每個不同算法在不同 threshold 下的宏觀預測準確值”，此時每個面積都考量到不同的 threshold，因此在這次作業不清楚資料分布的情形下，使用 auc score 這種相對宏觀的評斷方法可以使我們找出較好的模型。