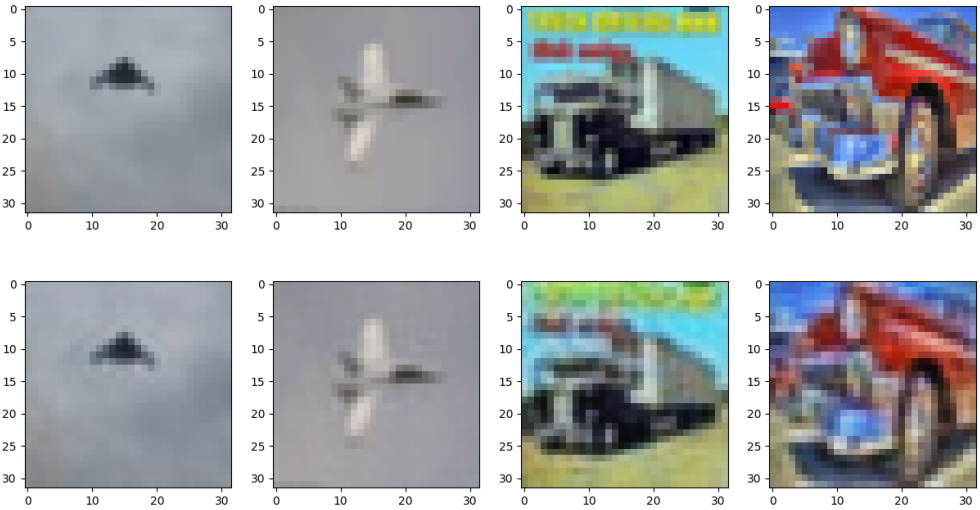
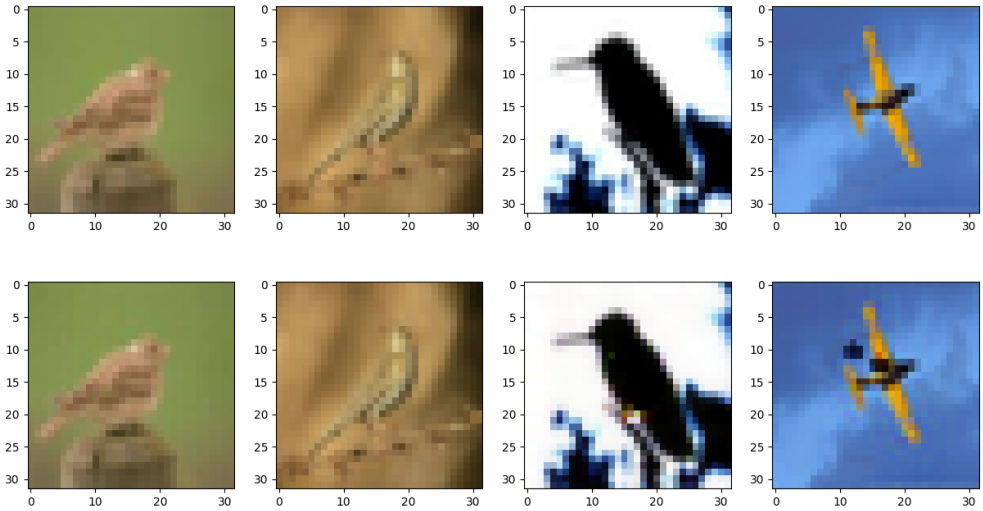
Machine Learning Spring 2020 - HW10 Report

學號: b07902064 系級: 資工二 姓名: 蔡銘軒

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，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。



上圖為sample code裡的cnn autoencoder ，經過101個 epoch 後所畫出來的圖。左到右分別是MSELoss 最小、次小、次大、最大，而上面一列為原圖，下面一列為重建的圖片。雖然圖片本身的像素就比較低，有些難以辨識，但是還是可以看到最左邊兩張圖，原圖與重建圖幾乎一模一樣，難以看出差異。而右圖兩張圖則可以看出差異。例如第三張的卡車，可以看出重建圖比原圖更加模糊，解析度感覺又更差了。第四張的重建圖也像是多了一層霧，且色調也跟原圖較為不同。



上圖是我的 best model 經過 101 個 epoch 後畫出來的圖。圖片的排列方式與 sample code model 相同。同樣的可以觀察到最左邊的兩張圖還原度非常高，基本上很難用肉眼看出差異。而第三張圖，雖然大部分的地方也還原得很好，但小鳥的腳的部分有一塊明顯的差異。最右邊的圖片則更明顯的沒有正確還原，有許多清楚可見的差異。

2. (1%) 嘗試把 sample code中的KNN 與 PCA 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。

```
(encoder): Sequential(
  (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): PReLU(num_parameters=32)
  (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (3): Conv2d(32, 32, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
  (4): PReLU(num_parameters=32)
  (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (6): Conv2d(32, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
  (7): PReLU(num_parameters=64)
  (8): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
(decoder): Sequential(
  (0): ConvTranspose2d(64, 32, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1))
  (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): ReLU()
  (3): ConvTranspose2d(32, 3, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1))
  (4): BatchNorm2d(3, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (5): Tanh()
)
```

上圖為我使用的autoencoder架構。訓練 101 個 epoch 後，在 kaggle 上的 AUC 分數為 0.70849。

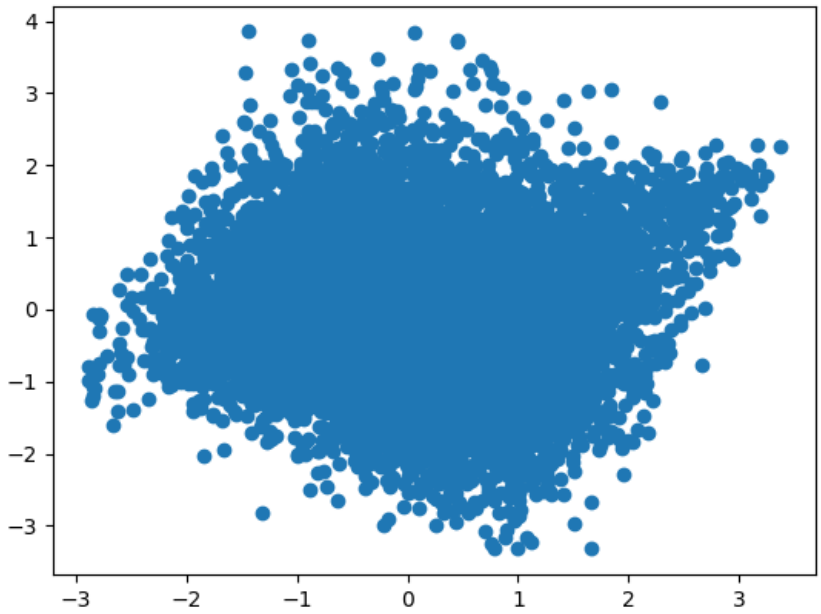
在 KNN 與 PCA 中，n_components 的數值我嘗試了7跟8。會選擇這兩個數字是因為在沒有 autoencoder 時，直接把 KNN 與 PCA fit 在 training data，然後對 testing data 做預測時，在這兩個數字我得到最好的結果。所以我猜測 training data 的資料可能是7群或是8群。

AUC score	KNN	PCA
n_components=7	0.59522	0.58008
n_components=8	0.59753	0.57913

從結果可以看到使用 KNN 與 PCA 後結果都變差了。我認為這有一些可能的原因，例如 KNN 以及 PCA 沒有辦法從 encoder 做出來的 latent 裡得到適合的資訊，而 decoder 因為是跟 encoder 一起訓練的，比較能解讀 latent 的資訊。另外這次作業裡 random 的成分很大。同樣的 autoencoder model，epoch 稍微多一點或少一點，或是使用不同的 random seed 初始化，得到的 AUC score 可以相差 0.1 以上。autoencoder 的結果是我經過非常多次嘗試才得到的，如果要改成跟 KNN 以及 PCA 配合，我認為還要經過很多微調才有機會做出更好的結果。

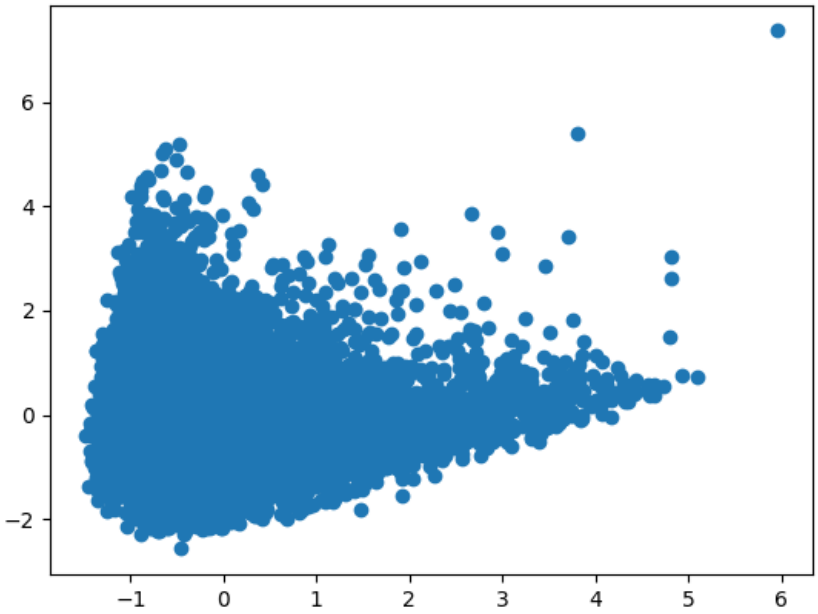
3. (1%) 如hw9，使用PCA或T-sne將testing data投影在2維平面上，並將testing data經第1題的兩顆model的encoder降維後的output投影在2維平面上，觀察經encoder降維後是否分成兩群的情況更明顯。

   PCA 直接對 testing data 降維:



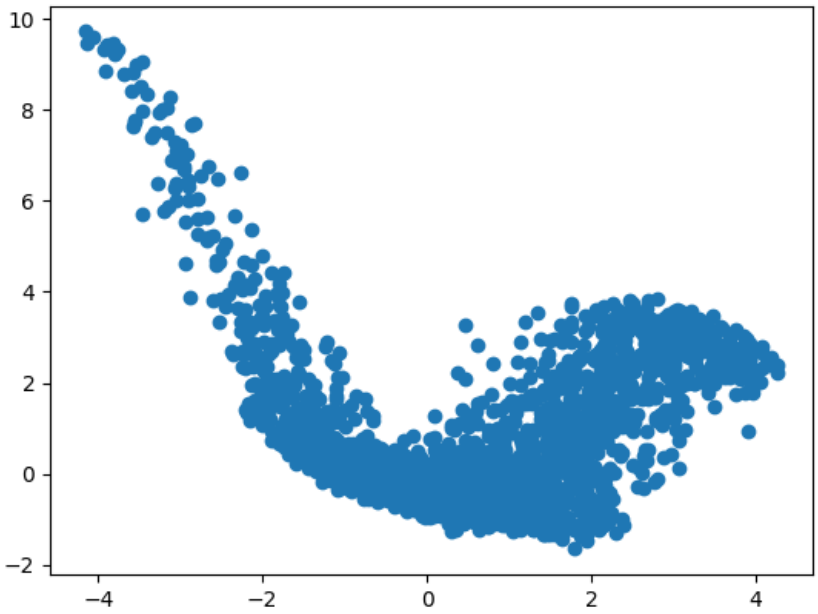
   直接對 testing data 降維並沒有使資料呈現明顯的分群，而是集中在中央形成一個大群。

   PCA 對 baseline model 的 latent 降維:



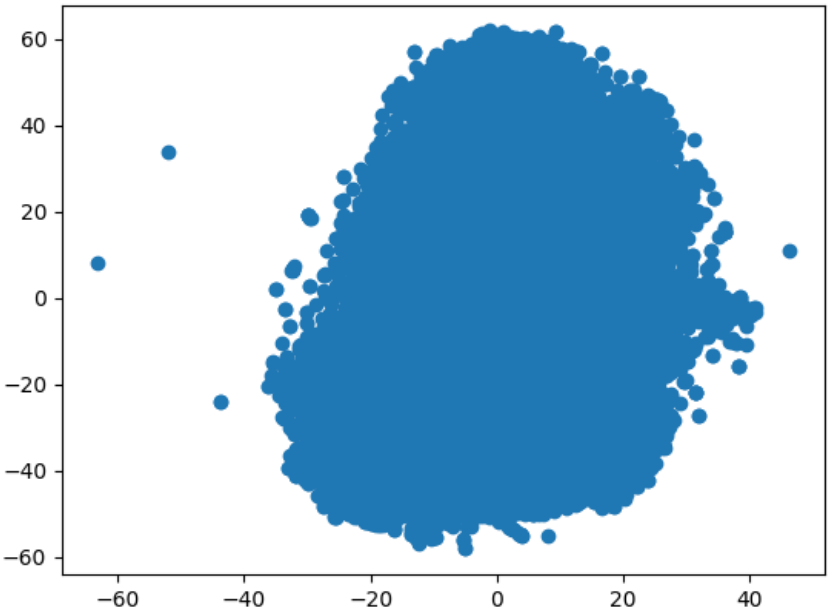
經過 baseline model 降維再投影到二維平面後，資料的分佈有所改變，但還是難以看出明顯的分群，不過可以看到零星幾個點分佈在右上半部，而不是全部都聚集在左下角。

PCA 對 best model 的 latent 降維:



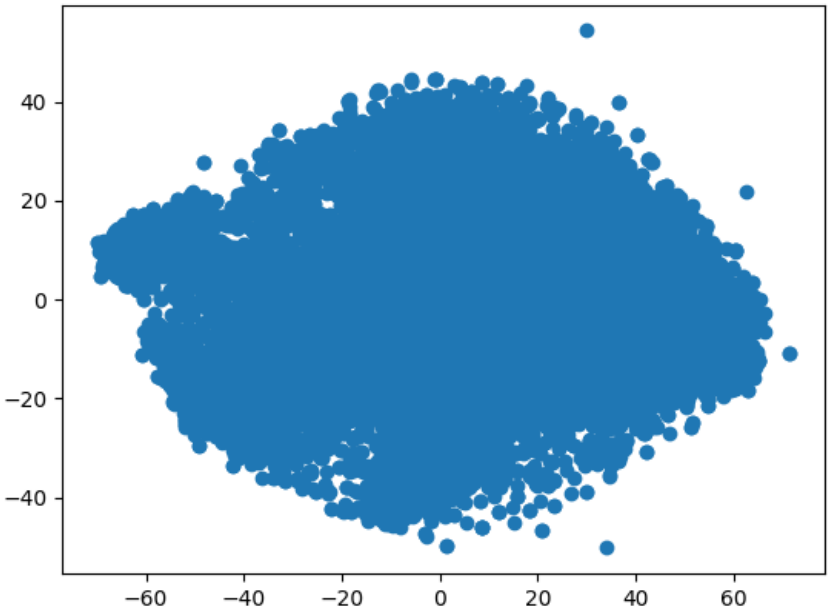
經過 best model 降維後再投影到二維平面，可以觀察到資料的分佈有比較不同的模式，不像先前兩張圖片大量集中在圖片裡的某個位置。這張圖片雖然還是沒有明顯的分群，但還是可以大致分為左半部較稀疏的點群跟右半部密集的點群。

TSNE 直接對 testing data 降維:



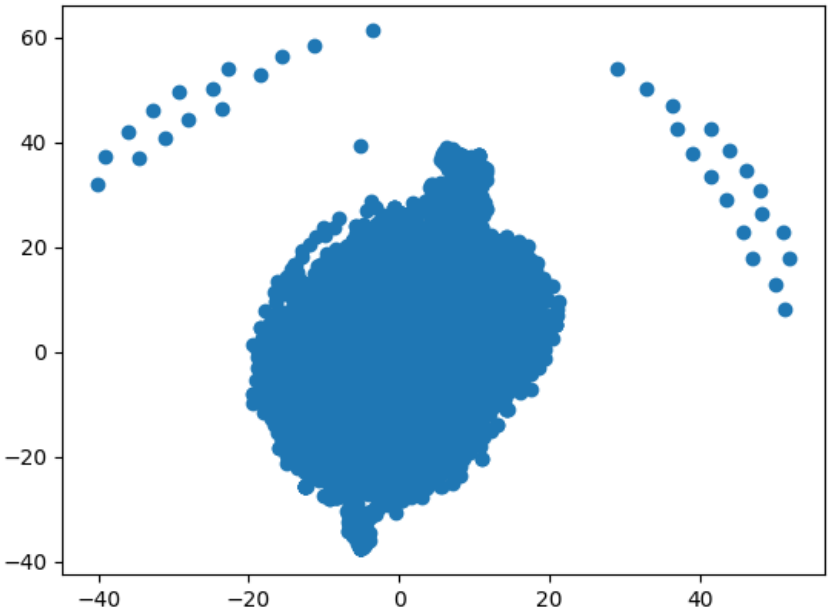
同樣地，直接對 testing data 降維後，資料密集的分佈在同一個地區，難以看出分群。

TSNE 對 baseline model 的 latent 降維:



雖然分佈略有改變，但依舊是大量集中在同一個地區，無法看出分群。

TSNE 對 best model 的 latent 降維:



這邊可以看到除了中間仍然有一群非常密集的點之外，在左右半部都分別有零星幾個點自成一群，算是在目前所有圖片裡，有比較明顯觀察到分群的圖片。

由 PCA 以及 TSNE 投影後，大部分還是難以看出 testing data 實際上到底分了幾群。我認為這個結果是因為許多 32x32x3 的圖片有許多資訊，很難在二維平面上就做到分群。如果想要觀察更明顯的分群，應該投影到更高維的空間才有機會將不同類型的圖片分開。

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？

若使用 f1 score，我們需要找到一個 threshold，將圖片分成兩類 (是否為anomaly) 才能判斷最後的分數。但我們使用不同方法會得到不同的分數分佈，且這些分佈都是連續的，很難找到一個切點來區分成兩類。如果要使用 f1 score，使用怎麼樣的 threshold 會很大的影響結果，且這也會變成一個需要調整與嘗試的參數，增加許多複雜的成分。

使用 ROC AUC 則沒有這個困擾，我們可以直接從每張圖片的分數算出最後的結果，不需要決定 threshold，因為 ROC AUC 的計算方式其實本身就考慮了各種不同的 threshold 了。