學號：B06901053　系級：電機三　姓名：謝承延

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）

Best Model: CNN
Kaggle AUC_ROC Score:0.55051　加上 PCA 後→0.60999
總參數量:47355
架構:

	Type	Filter Size	Input size
Encoder	Conv2D	4x4x3x12	32x32x3
	ReLU		16x16x12
	Conv2D	4x4x12x24	16x16x12
	ReLU		8x8x24
	Conv2D	4x4x24x48	8x8x24
	ReLU		4x4x48
Decoder	ConvTranspose2D	4x4x48x24	4x4x48
	ReLU		8x8x24
	ConvTranspose2D	4x4x24x12	8x8x24
	ReLU		16x16x12
	ConvTranspose2D	4x4x12x3	16x16x12
	Tanh		32x32x3

相關參數:

Batch size:128
Learning rate:0.001
Epoch :5000
Optimizer: AdamW

Train Loss: 0.0025

可以看到使用 AdamW 做訓練收斂非常快，大約 1000 個 epoch 以後就不再變動了。

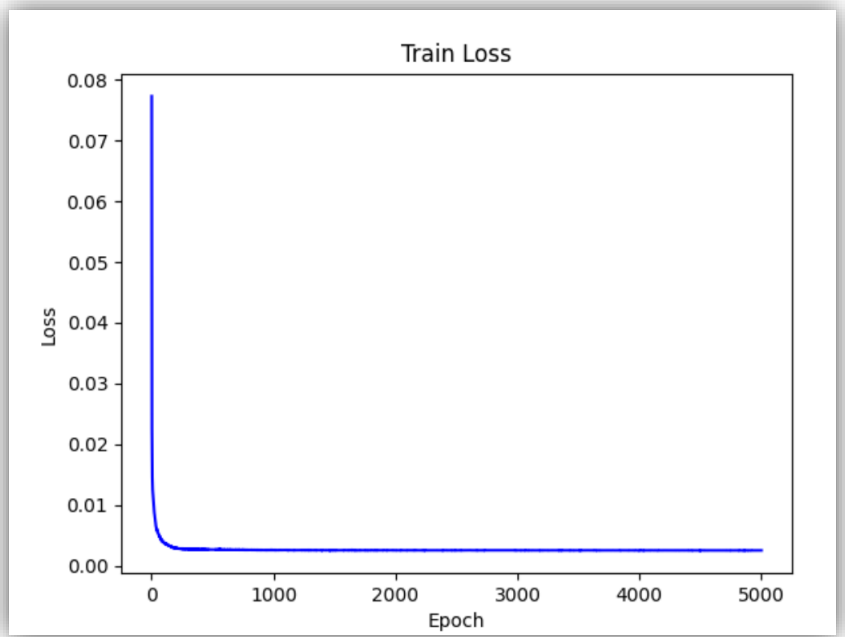


Image with largest MSE 0.017434

左為原圖，右為 Reconstructed image

可以看到右上角的顏色比原圖來的模糊。

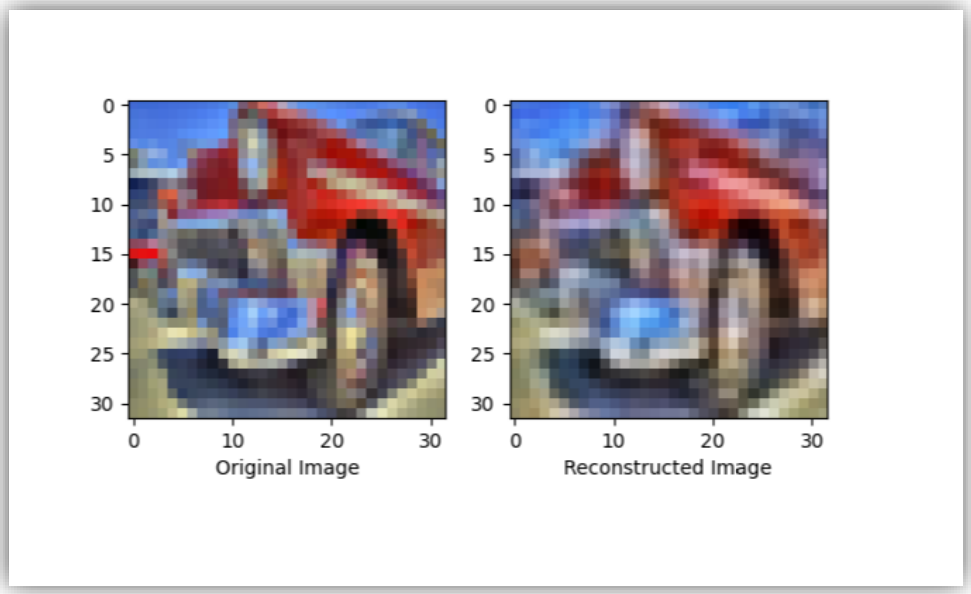


Image with 2-nd largest MSE 0.016123

右方為 Reconstructed，可以看到樹枝部分跟原圖相差較多。

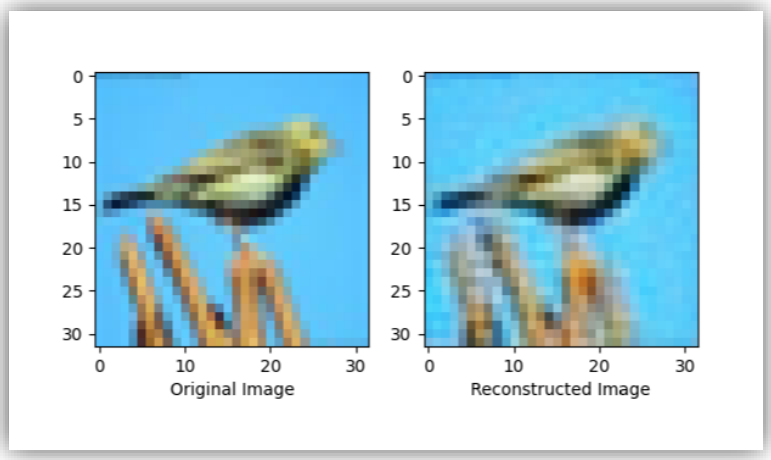


Image with smallest MSE 0.000142

重建後的 image 跟原本的 image 大致相同。

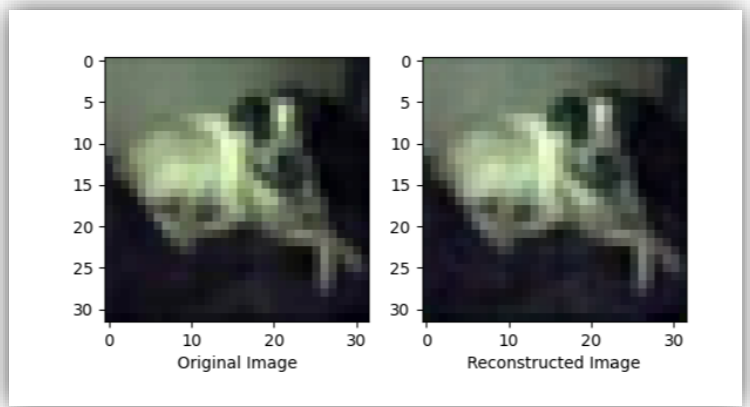
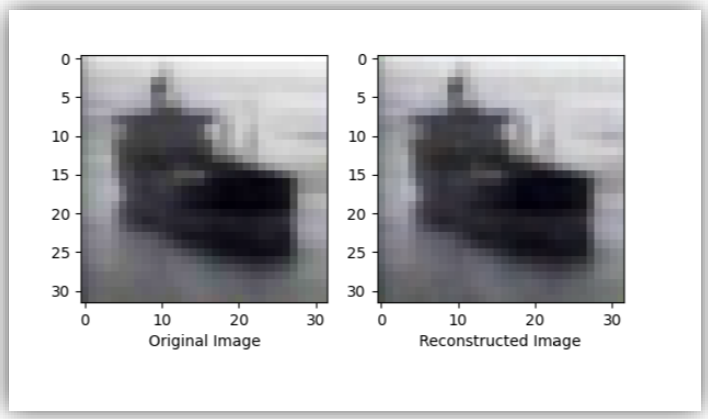


Image with 2-nd smallest MSE 0.000151



Baseline Model: FCN

總參數量:807907

Kaggle AUC_ROC Score:0.59565

架構:

	Type	Size
Encoder	Linear	(32*32*3,128)
	ReLU	
	Linear	(128,64)
	ReLU	
	Linear	(64,12)
	ReLU	
	Linear	(12,3)
	ReLU	
Decoder	Linear	(3,12)
	ReLU	
	Linear	(12,64)
	ReLU	
	Linear	(64,128)
	ReLU	
	Linear	(128,32*32*3)
	Tanh	

相關參數:

Batch size:128

Learning rate:0.001

Epoch :1000

Optimizer: AdamW

Train Loss: 0.118014

Image with largest MSE 0.542

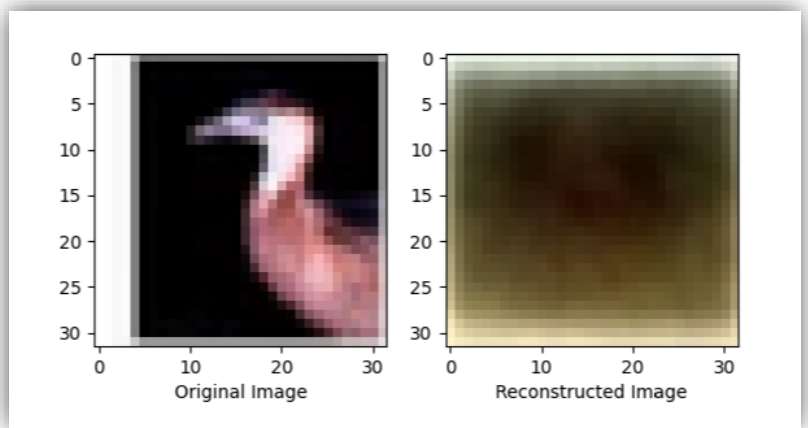


Image with 2-nd largest MSE 0.509

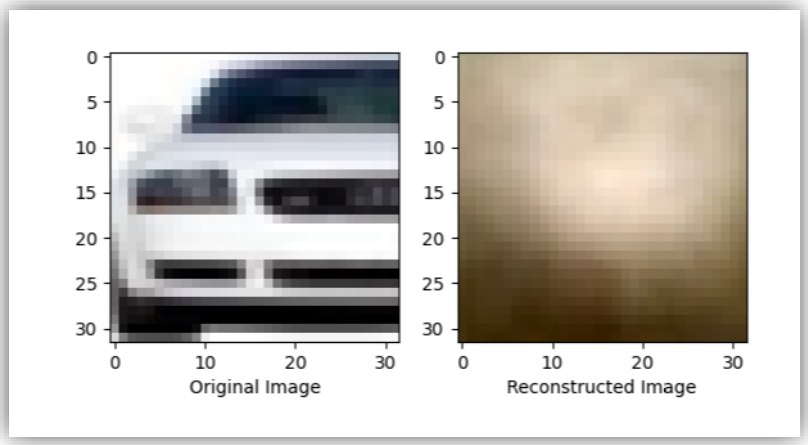


Image with smallest MSE 0.009

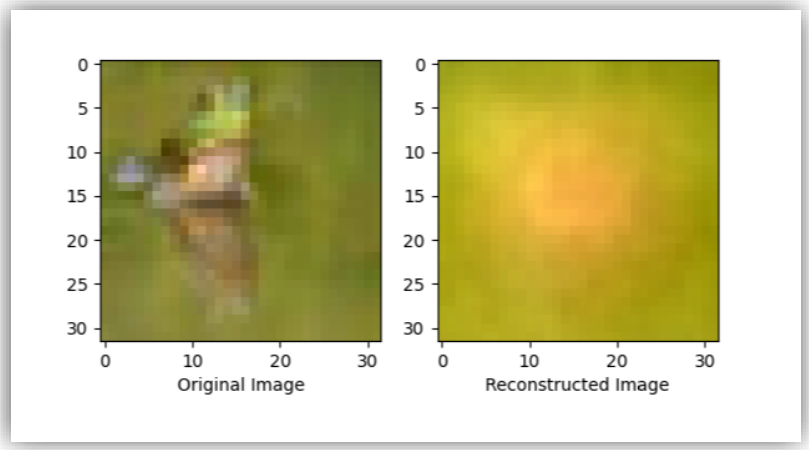
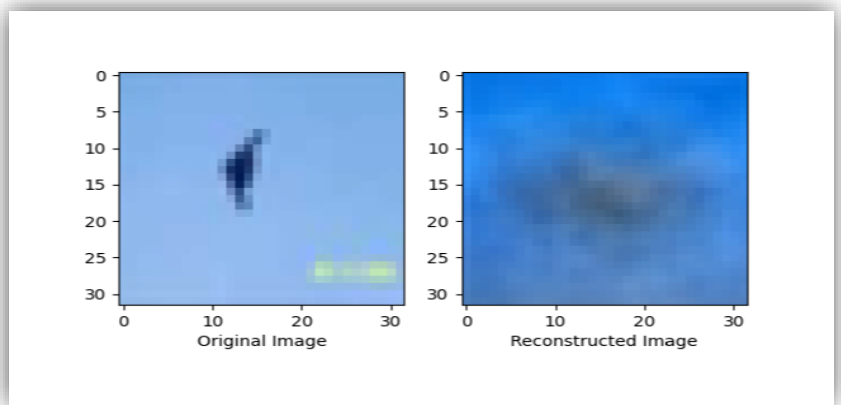


Image with 2-nd smallest MSE 0.010



以上的結果是單純看 reconstruct 後得到的 MSE。
可以看到在 kaggle 上的表現 FCN 是比較好的，然而在重建 image 的能力上 CNN 好出許多，FCN 重建的圖片幾乎每一張都看不出來原圖到底是甚麼，CNN 重建的 image 中即使 MSE 最大的圖片也跟原圖非常相像，這有可能是因為 CNN 對於圖片分析的能力較強，所以即使是異常的圖片，他也能夠成功重建，但是這也導致 CNN 若單以 reconstruct 的圖片跟原圖的 MSE 來分析異常狀況效果會比較差。
FCN 將圖片完全變成一維來處理，可能分析圖片的能力會較差，因此碰到異常狀況就會無法正常重建，導致 MSE 較高，這或許就是 FCN 能獲得較高 AUC_ROC score 的原因。
1.為何 CNN 能夠重建圖片的較好:
可能的原因是:CNN 本身的 feature extraction 能力較好，即使面對沒有看過的圖片，也有足夠能力取到有用的 feature，讓 decoder 能夠重建回原本的圖片。
2.為何單用 CNN 的結果較差:
這個問題可能的原因大概跟為何能重建好圖片是相關的，CNN 本身分析圖片的能力太好，使得沒有看過得圖片也能夠被很好的重建，然而對於 Anomaly Detection 這個任務而言，沒看過的圖片應該要無法被重建，才能讓我們正確判斷何為異常。
第二個原因是 CNN 的 Dimension Reduction 做得較少，可以看到 CNN 架構 bottleneck 的部分 size 為 4x4x48=768，input size 為 32x32x3=3072，兩者僅差 4 倍，也就是說，可能因為維度下降的不夠，讓 bottleneck 含有的資訊過多，使得異常圖片也能夠被 reconstruct。
3.使用 PCA:
CNN 在使用 PCA 於 Encoder Output 之後效果有明顯的改善，可能的原因大概跟上述相同，就是 Dimension Reduction，假設 PCA 所做的 Dimension Reduction 是將 feature 資訊壓縮到更小的維度形成 latent，當壓縮的維度夠小，異常圖片可能就會因壓縮後的 latent 資訊不夠而無法 reconstruct 圖片。
4.可能的改善方向:
實驗過幾次之後，我發現把 CNN 的架構改成知名的架構如 VGG16 ResNet 等反而會使結果變差，可見這些架構分析圖片能力太好，對於異常圖片也能夠重建，所以難以判斷哪些是異常圖片。
因此我認為，或許可以將 CNN 的架構套用在 VAE 中，原 sample Code 中的 VAE 是使用 FC 的架構，分析圖片的能力較差，encoder 跟 decoder 若改為 convolution layers，可能可以讓 Reconstruction MSE 變小，另外 VAE 的 loss 除了由 MSE 組成，還有一項 KL Divergence，這一項會讓 encoder 求出的σ跟υ盡量靠近 0(為了讓 z 更像 normal distribution)，我們也許可以假設異常圖片會使得 Encoder 產生出來的 z 很不像 normal distribution，使得 Reconstruct 的圖片 MSE 較高。

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。

此題中我使用的 AutoEncoder 為第一題所用的 CNN。

Model: CNN

Kaggle AUC_ROC Score:0.55051

總參數量:47355

架構:

	Type	Filter Size	Input size
Encoder	Conv2D	4x4x3x12	32x32x3
	ReLU		16x16x12
	Conv2D	4x4x12x24	16x16x12
	ReLU		8x8x24
	Conv2D	4x4x24x48	8x8x24
	ReLU		4x4x48
Decoder	ConvTranspose2D	4x4x48x24	4x4x48
	ReLU		8x8x24
	ConvTranspose2D	4x4x24x12	8x8x24
	ReLU		16x16x12
	ConvTranspose2D	4x4x12x3	16x16x12
	Tanh		32x32x3

相關參數:

Batch size:128

Learning rate:0.001

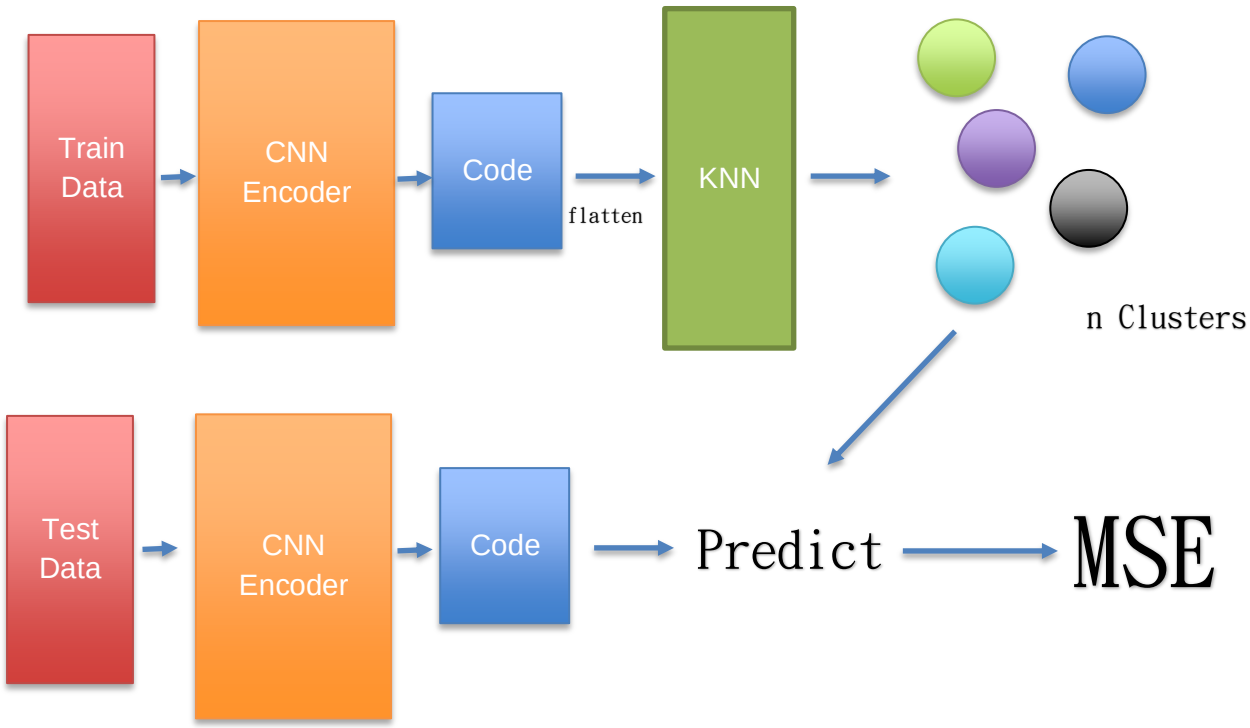
Epoch :5000

Optimizer: AdamW

實作方法: Encoder + KNN

我直接將 Report1 訓練好的 CNN model 拿來用，將 AutoEncoder 的 Encoder Output 取出後攤平丟入 KNN，首先先將 train_data 的丟入得到 encoder output，以下稱之為 train_output，將 train_output 放入 KNN 做 clustering 後，將 test_data 也丟入得到 Encoder output(以下稱之 test_output)，並用前面 train_output clustering 的結果去預測 test_output 屬於哪一 cluster，最後以這些預測結果算出 testing data 的 MSE。

圖示:

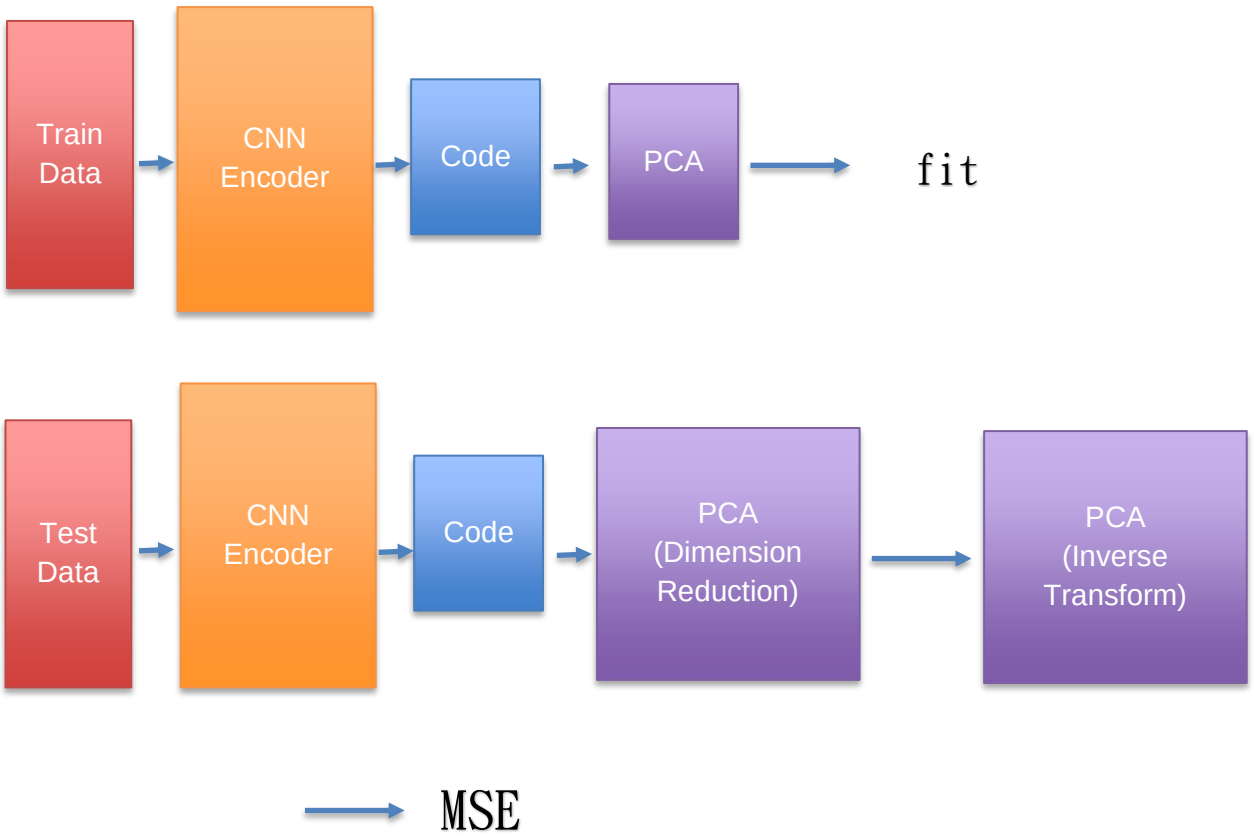


相關參數與結果:

Number of cluster	Kaggle AUC_ROC Score	Original Score
10	0.60939	0.55051
5	0.60856	0.55051

實作方法: Encoder + PCA

將 Report1 訓練好的 CNN model 拿來用。首先將 train data 丟入 AutoEncoder 的 Encoder Output 取出，攤平後丟入 PCA，PCA（ z = Wx ）會利用此 Encoder output 的資訊完成 fit，接著將 test data 丟入 AutoEncoder 的 Encoder output 取出，攤平後丟入 PCA，PCA 會把該 output 減維，然後再 inverse transform 回原本的維度，此時就可以利用未丟入 PCA 的 output 跟丟入 PCA 後的結果計算 MSE。



相關參數與結果:

N_component	Kaggle AUC_ROC Score	Original Score
2	0.60999	0.55051

觀察:

可以發現比起原本的 AutoEncoder, 使用 PCA 跟 KNN 得到的效果是更好的，這可能是因為 CNN 本身的 Generalize 能力較強，在面對沒看過的 data 的時後仍然能夠將圖片重建，因此使得異常狀況辨識能力不佳，PCA 將 data 投射到低的維度，如果是異常 data 在低維度就會被丟開，KNN 則是利用分群來辨識出異常 data，以此實驗結果看起來使用這兩種方法都有所幫助。

1.KNN 跟 PCA 的差異:

以 CNN 架構而言，他的 bottleneck 是 4x4x48=784 維度的資料，KNN 是將這 784 維度分群，PCA 則是將這 784 維度的資料做減維。

2.為何 KNN 跟 PCA 都取得了不錯的結果:

原本單用 CNN 的時候結果非常爛，用了 KNN 跟 PCA 後結果都有所改善，可見 CNN 找出來的 latent 具有差異性，可以通過對 latent 做處理而得到更好的結果，而原本單純看重建圖片 MSE 的話，因為 CNN 本身重建圖片能力太好，連異常圖片都能夠重建的很好，就很難判斷異常圖片。

3.在 FCN 使用 PCA:

我也在 FCN 嘗試使用 PCA，發現結果完全爛掉，我認為可能的原因是，FCN 的架構產生的 latent 全部都攪在一起了，此時當我們去做 PCA/KNN 可能會把異常圖片的 latent 跟正常圖片的 latent 弄在一起，如此一來就無法判斷異常圖片了。

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）

Best Model: CNN

總參數量:47355

架構:

	Type	Filter Size	Input size
Encoder	Conv2D	4x4x3x12	32x32x3
	ReLU		16x16x12
	Conv2D	4x4x12x24	16x16x12
	ReLU		8x8x24
	Conv2D	4x4x24x48	8x8x24
	ReLU		4x4x48
Decoder	ConvTranspose2D	4x4x48x24	4x4x48
	ReLU		8x8x24
	ConvTranspose2D	4x4x24x12	8x8x24
	ReLU		16x16x12
	ConvTranspose2D	4x4x12x3	16x16x12
	Tanh		32x32x3

相關參數:

Batch size:128

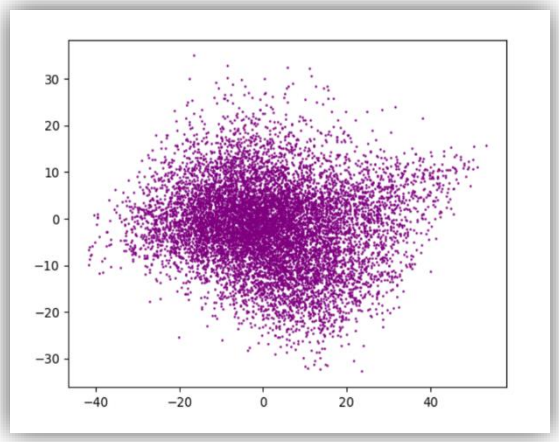
Learning rate:0.001

Epoch :5000

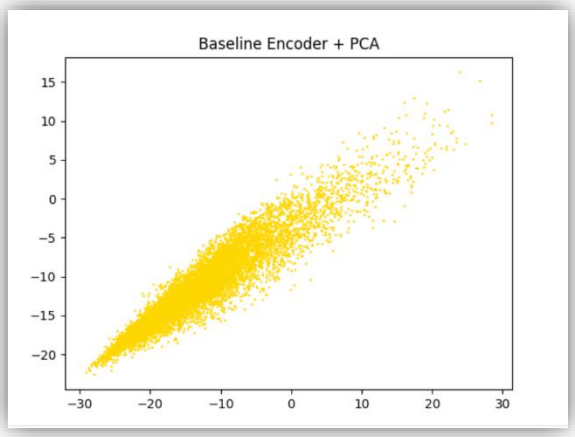
Optimizer: AdamW

結果:

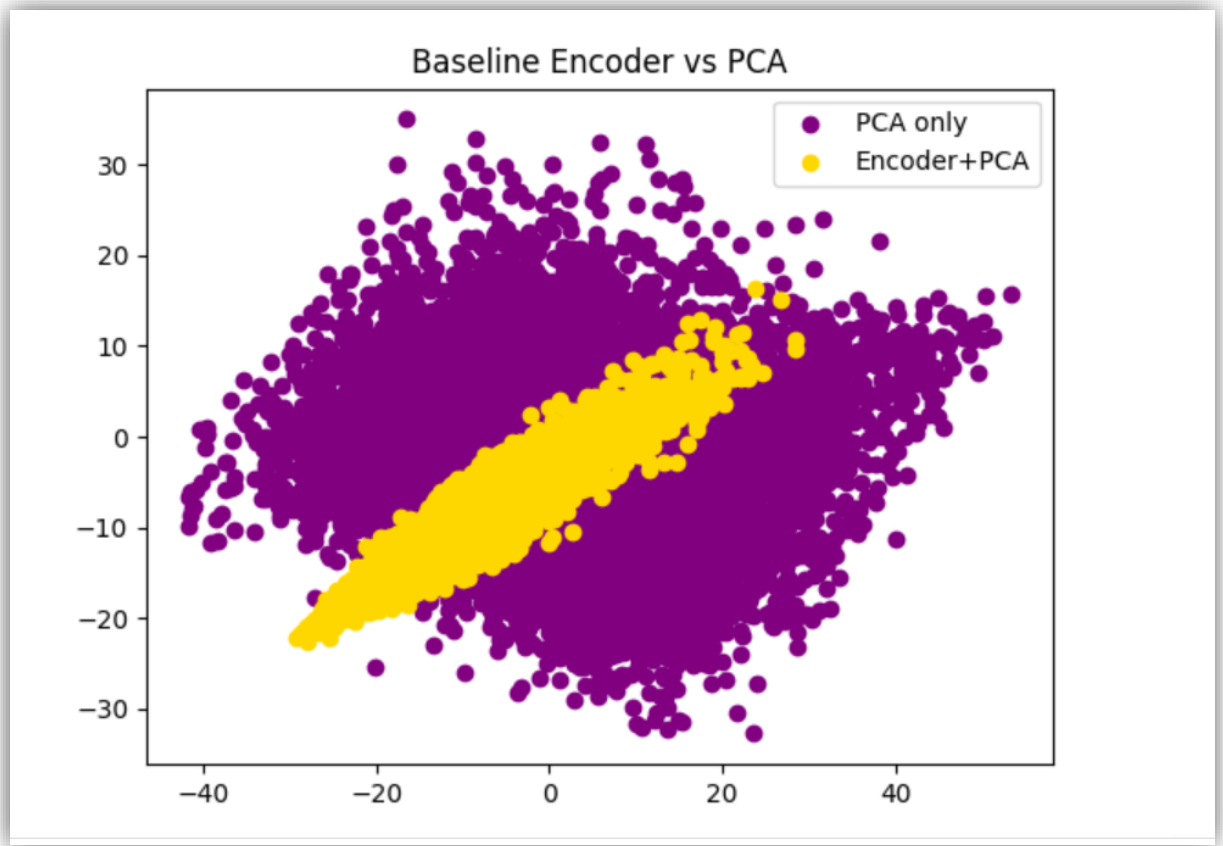
PCA 投影結果:



Encoder 降維+PCA:



比較:



黃色為 Encoder 降維後使用 PCA 投影到二維，紫色為單純使用 PCA，兩者皆不太能看出有明顯分成兩群的情況，然而使用 Encoder 降維的情況投影的結果較集中，這或許是個好現象，因為這樣出現異常狀況時，他的 MSE 就容易變得比較大，也就是說比較容易偵測到異常狀態。

Baseline Model: FCN

總參數量:807907

Kaggle AUC_ROC Score:0.59565

架構:

	Type	Size
Encoder	Linear	(32*32*3,128)
	ReLU	
	Linear	(128,64)
	ReLU	
	Linear	(64,12)
	ReLU	
	Linear	(12,3)
	ReLU	
Decoder	Linear	(3,12)
	ReLU	
	Linear	(12,64)
	ReLU	
	Linear	(64,128)
	ReLU	
	Linear	(128,32*32*3)
	Tanh	

相關參數:

Batch size:128

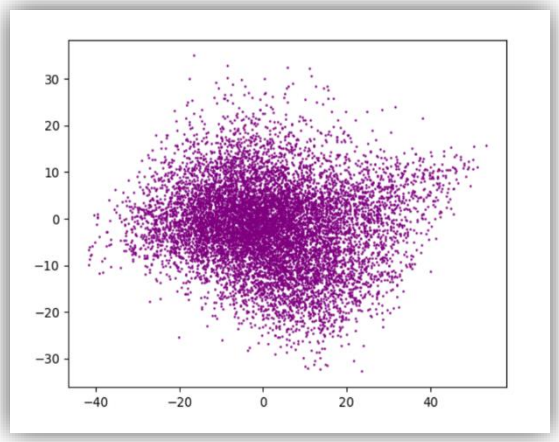
Learning rate:0.001

Epoch :1000

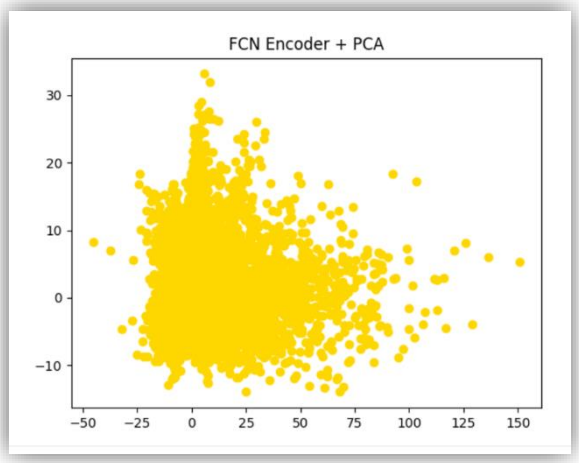
Optimizer: AdamW

結果:

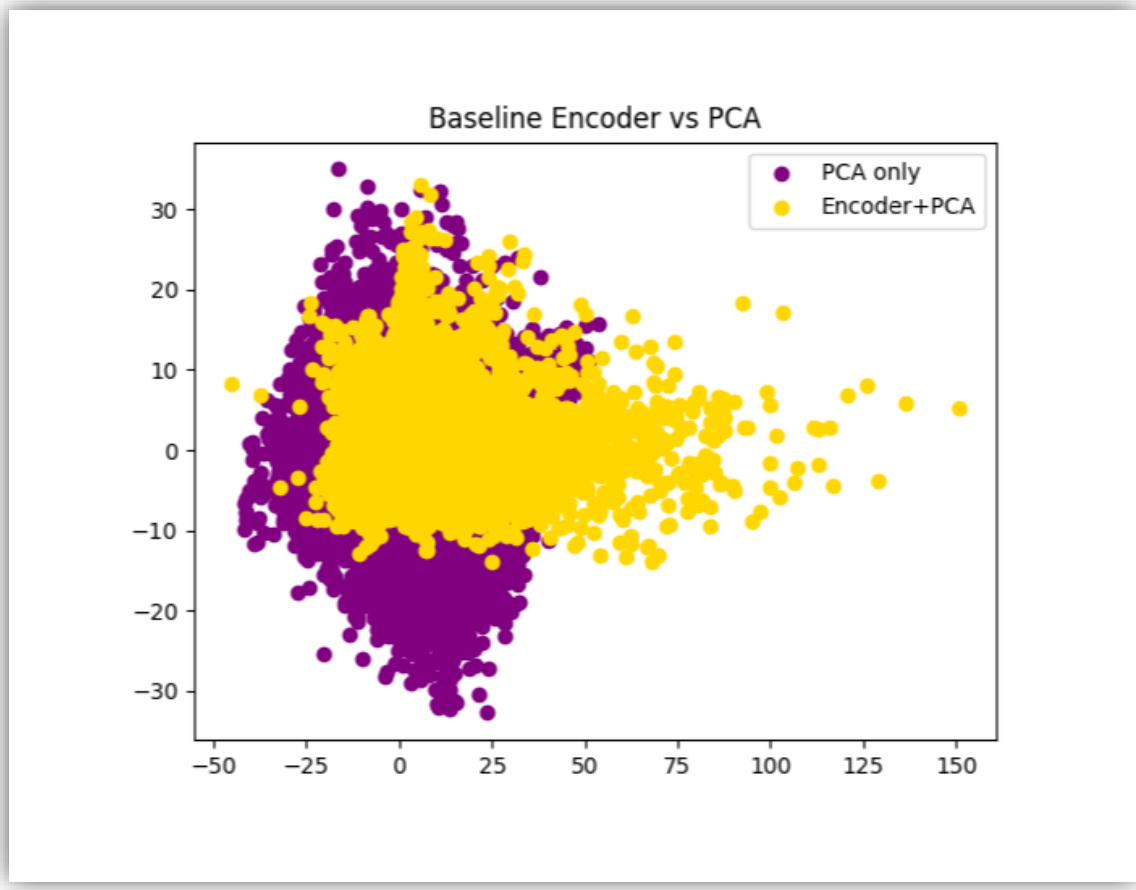
PCA 投影結果:



Encoder 降維+PCA:



比較:



黃色:Encoder 降維後用 PCA
紫色:單純使用 PCA
可以發現使用 FCN 並用 PCA 投影到二維的狀況中，點分布的比較分散，這代表 Encoder 可能有確實幫助到 PCA 做分群的動作。

```
y_projected = pca.fit_transform(reconstructed)
y_reconstructed = pca.inverse_transform(y_projected)
dist = np.mean(np.square(y_reconstructed - reconstructed).reshape(len(y), -1), axis=1)
```

然而當我使用 FCN，取出 Encoder output 後做 PCA 的 fit_transform，並將 PCA 前後的取 MSE 時(上圖的 reconstructed 是單純 encoder 的 output，而非 reconstructed image)，即使最大 MSE 之間的區別仍然很明顯，我卻發現 Kaggle 上的表現完全爛掉了…，我認為原因可能是 FCN+PCA 錯把一些正常的 data 投影到偏差很大的位置，變成正常 data MSE 很大，異常 data 反而 MSE 很小。

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？

Reference: https://blog.csdn.net/shenxiaoming77/article/details/72627882

首先需要先描述兩種衡量標準。

實際值	預測值	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

T 代表 True 也就是預測成功

F 代表 False 也就是預測錯誤

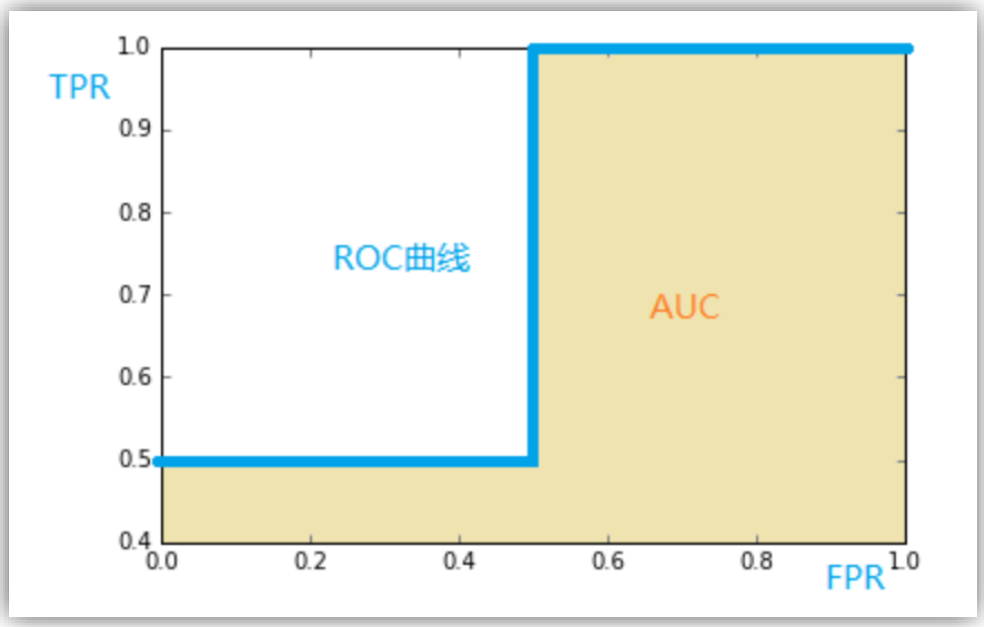
P 代表 Positive 代表預測為正常

N 代表 Negative 代表預測為異常

Auc Score:

$$\text{TPR} = \frac{\text{TP}}{(TP + FN)}$$

$$\text{FPR} = \frac{\text{FP}}{(FP + TN)}$$



這個衡量方式是通過計算 FPR 跟 TPR 畫出 ROC 曲線，其中在 ROC 曲線下的總面積就叫做 AUC(Area Under the Curve)。

計算過程為:

提供預測的 score 跟真實 label，接著把這些 score 做排序，每個 score 輪流有小到大做為 threshold，只要超過這個 threshold 就會被判定為 positive，舉例來說假設預測的 score 為[0.1 0.4 0.35 0.8] 實際 y 值為[P N P N]
則我們會輪流以 0.1 0.35 0.4 0.8 做為 threshold，假設我們以 0.1 做為 threshold，就會得到

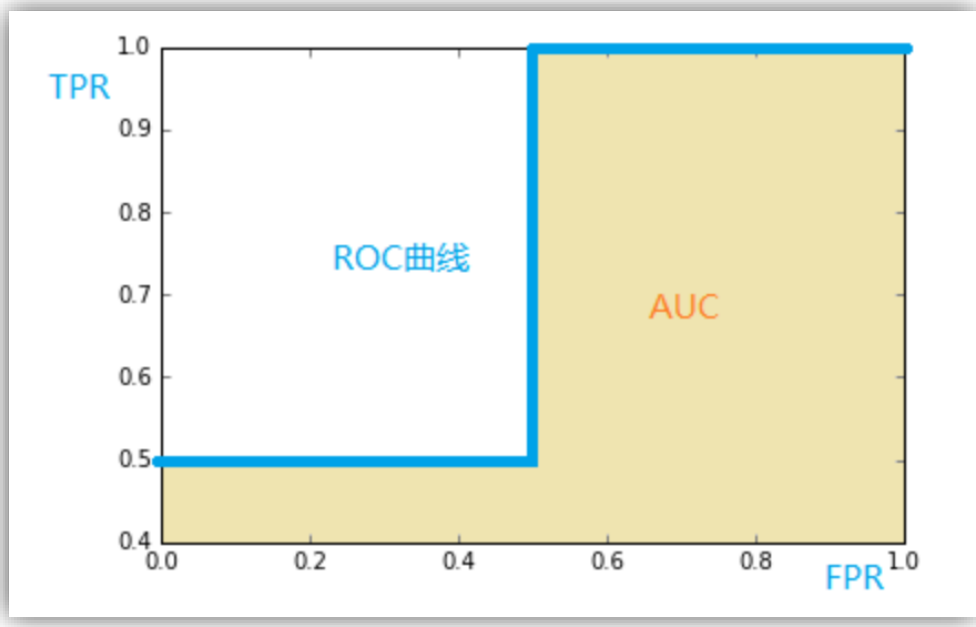
$$y\_true = [P\ N\ P\ N]$$
$$y\_pred = [P\ P\ P\ P]$$

可得到

實際值	預測值	
	Positive	Negative
Positive	2	0
Negative	2	0

可由此計算出 TPR=1 FPR=1。

重複以上過程使用不同 score 當作 threshold 後可得到很多組(FPR,TPR)，以 FPR 做為 x 座標，TPR 做為 y 座標，並將這些點畫出來連線就可以得到 ROC 曲線，如此一來便能得到 AUC。



F1_score:

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})}$$

$$\text{Recall} = \frac{\text{TP}}{(TP + FN)}$$

這個方法則是計算 precision 和 Recall 去得到 score。
Precision 相當於判斷為 positive 時有多少信心預測對，Recall 則是實際為 Positive 的時後有多少機會判斷對。
這個衡量標準的好處是可以處理 precision recall 之間的平衡，因為一般 Precision 好的 model，Recall 就不會太好，反之 Recall 好的 model，Precision 就沒有太好。

比較:
為何使用 AUC_Score?

觀察到 F1_score 的式子，我們可以發現到，他主要的重點是 focus 在 TP 的預測能力上，然而他卻沒有考慮到偵測異常能力(例如 TN 跟 FP 的比例)的狀況，也就是當這個 case 實際值為 Negative 的時候，我們有多少信心能夠判斷出來。

反之，AUC_Score 則有考慮到這樣的能力，我們可以注意到 FPR 的式子

$$\text{FPR} = \frac{\text{FP}}{(FP + TN)}$$

也就是當實際值為 Negative 的時候，被預測成 Positive 的機率有多少(所以一般會希望 FPR 越低越好。)
平常我們做 binary classification 的時候，我們比較在乎的是這個 model 他正確分類的能力有多少，意即我們會比較在乎 TP，此時用 F1_score 就比較恰當，然而在做 Anomaly Detection 的時候，偵測異常的能力也是很重要的，因 F1_score 的公式較少這方面的判斷標準，使用 AUC_Score 可能就變成較好的選擇。

F1_score 的不便之處:
除了上述提到 F1_score 的問題之外，F1_score 還必須手動選擇 threshold，一般做預測的時候，每個 test_data 都會有一個 score，若 score 超過一定的 threshold 就會被判斷為 positive，AUC_score 的做法當中，直接利用由小到大排序的 score 來當作 threshold，F1_score 則需要自己判斷，代表 threshold 的選擇可能會對 F1_score 的結果有較大的影響性。
反之 AUC_Score 利用所預測的 score 當作 threshold 的好處是:得出的結果比較平均。