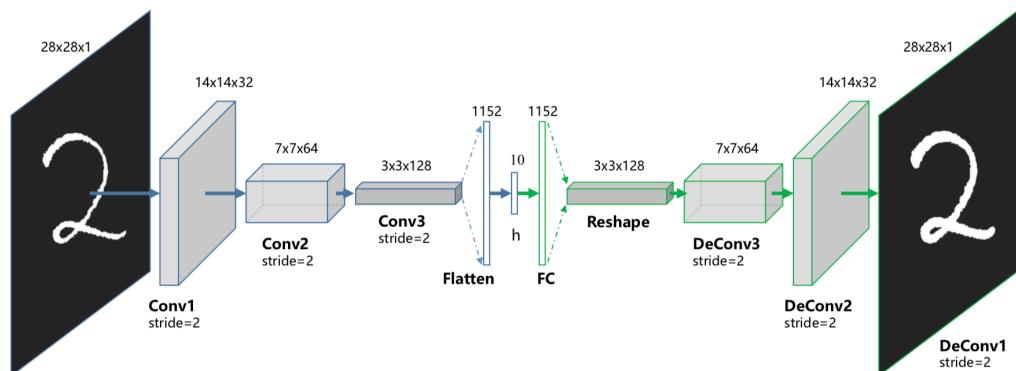


1. 請使用不同的 Autoencoder model，以及不同的降維方式(降到不同維度)，討論其 reconstruction loss & public / private accuracy。（因此模型需要兩種，降維方法也需要兩種，但 clustering 不用兩種。）

不同 Model 比較：

Convolution 完加入 Fully connected	Convolution 完沒有加入 Fully connected
<pre>class Autoencoder(nn.Module): def __init__(self): super(Autoencoder, self).__init__() self.encoder = nn.Sequential(#32*32 nn.Conv2d(3, 32*C, 3, 2, 1), #16*16 nn.SELU(), nn.Conv2d(32*C, 64*C, 3, 2, 1), #8*8 nn.SELU(), nn.Conv2d(64*C, 128*C, 3, 2, 1), nn.SELU()) # define: decoder self.decoder = nn.Sequential(nn.ConvTranspose2d(128*C, 64*C, 2, 2), nn.ConvTranspose2d(64*C, 32*C, 2, 2), nn.ConvTranspose2d(32*C, 3, 2, 2), nn.Tanh()) self.linear_i=nn.Linear(z_input,hidden) self.linear_o=nn.Linear(hidden,z_input) def forward(self, x): encoded = self.encoder(x) encoded=self.linear_i(encoded.view(encoded.size(0), -1)) #print(encoded.shape) hold=self.linear_o(encoded) decoded = self.decoder(hold.reshape(-1,128,4,4)) return hold, decoded</pre>	<pre>class Autoencoder(nn.Module): def __init__(self): super(Autoencoder, self).__init__() self.encoder = nn.Sequential(#32*32 nn.Conv2d(3, 256, 3, 2, 1), #16*16 nn.SELU(), nn.Conv2d(256,128, 3, 2, 1),#8*8 nn.SELU(), nn.Conv2d(128,64, 3, 2, 1), #4*4 nn.SELU()) self.decoder = nn.Sequential(nn.ConvTranspose2d(64,128, 2, 2), nn.ConvTranspose2d(128,256, 2, 2), nn.ConvTranspose2d(256, 3, 2, 2), nn.Tanh()) def forward(self, x): encoded = self.encoder(x) decoded = self.decoder(encoded) return encoded, decoded</pre>
Public	0.81518
Private	0.81190
reconstruction loss	0.0017



架構參考自 Deep Clustering with Convolutional Autoencoders(Xifeng Guo2017)

Encoder 和 Decoder 再經過 convolution 完之後，加入 fully connected layer，可以發現不論是 Publicc 還是 Private 分數都上升很多。此外，還可以發現 **reconstruction loss** 有經過 Fully connected layer，會比較難 reconstruct 回原本的圖型，其 reconstruction loss 會比較高，但也是因為這樣不論是 Publicc 還是 Private 分數都上升很多。

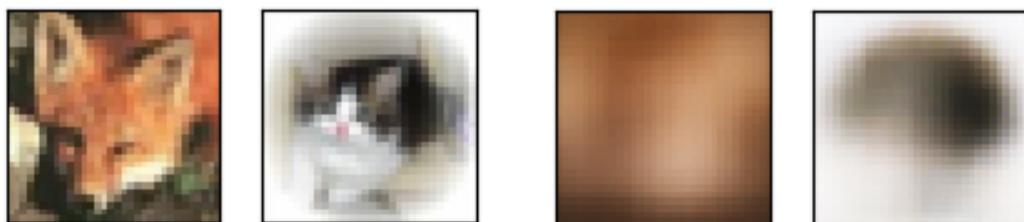
不同維度之比較：

降維至 12 維		降維至 128 維
Public	0.80555	0.72333
Private	0.80285	0.73126
reconstruction loss	0 . 00006	0 . 00001

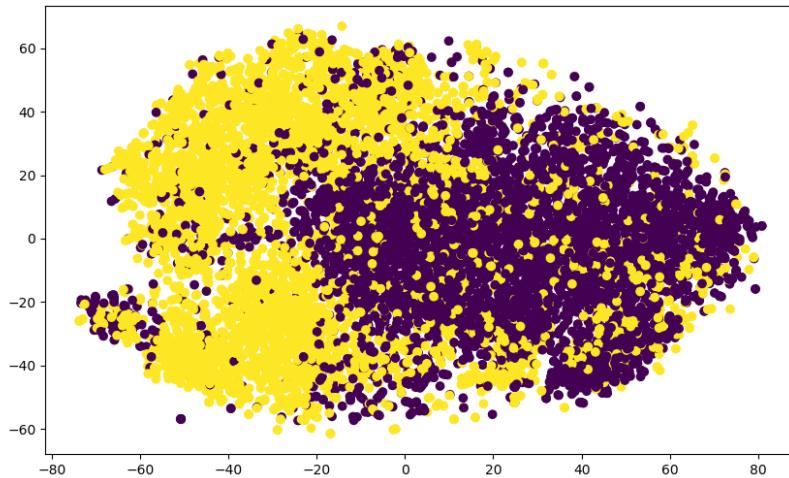
可以發現降維至 128 維 **reconstruction loss** 會降低很多，因為降至 128 維比較好把原本的 data reconstruct 回來，不過 Publicc 還是 Private 分數都會比降維至 12 維來的低。所以可以發現在這一個 task 中，找到比較重要的那幾個 eigen vector 會比，使用全部的 eigen vector 來的重要很多。此一現象可以在 Publicc 還有 Private score 發現。

2. 從 dataset 選出 2 張圖，並貼上原圖以及經過 autoencoder 後 reconstruct 的圖片。

可以發現大致可以把圖片的的大致色塊解回來，但卻解不回清晰的輪廓以及邊界。

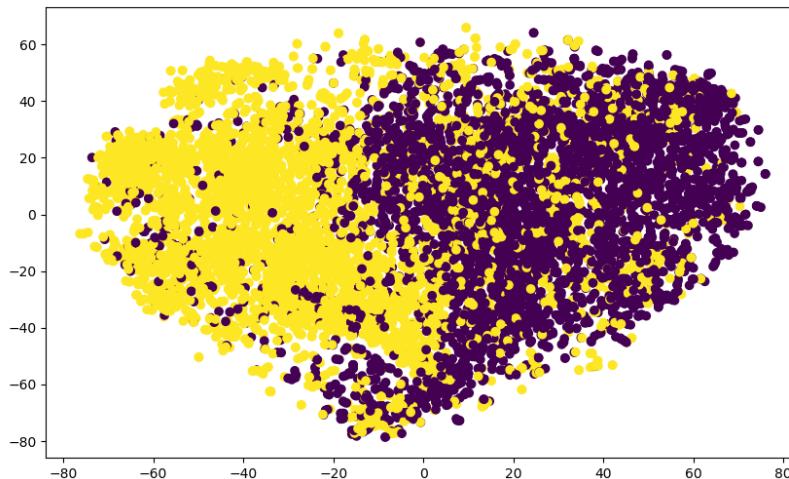


3. 在之後我們會給你 dataset 的 label。請在二維平面上視覺化 label 的分佈 Cluster 之前的：



Cluster 之後的：

可以發現再分類之後，黃色會往左邊集中，紫色會往右邊集中。



4. Principle Component Analysis (1%)

Given 10 samples in 3D: (1, 2, 3), (4, 8, 5), (3, 12, 9), (1, 8, 5), (5, 14, 2),
(7, 4, 1), (9, 8, 9), (3, 8, 1), (11, 5, 6), (10, 11, 7)

(a) principal axes:

$$\lambda_1, \lambda_2, \lambda_3 = (15.2974434)(11.63052369)(5.47203291)$$
$$\begin{pmatrix} 0.6165947 \\ 0.58881629 \\ 0.52259579 \end{pmatrix} \begin{pmatrix} 0.67817891 \\ -0.73439013 \\ 0.02728563 \end{pmatrix} \begin{pmatrix} 0.39985541 \\ 0.33758926 \\ -0.85214385 \end{pmatrix}$$

(b) Please compute the principal components for each sample.

在算這一題時，會發現用程式親自算和 sklearn 的 pca 算出不一樣的答案，主要原因在於 PCA 在計算時，會先對資料平移平均數，所以 pca 解出來的 principal component 才會和手算的有誤差。

principal components for each sample:

$$\begin{pmatrix} 3.36201464 \\ -0.70874446 \\ -1.48139761 \end{pmatrix} \begin{pmatrix} 9.78988804 \\ -3.02597728 \\ 0.03941652 \end{pmatrix} \begin{pmatrix} 13.61894165 \\ -6.53257419 \\ -2.41865723 \end{pmatrix} \begin{pmatrix} 7.94010395 \\ -5.06051399 \\ -1.16014972 \end{pmatrix} \begin{pmatrix} 12.37159312 \\ -6.83599606 \\ 5.02123906 \end{pmatrix}$$

$$\begin{pmatrix} 7.19402383 \\ 1.83697744 \\ 3.29720109 \end{pmatrix} \begin{pmatrix} 14.96324467 \\ 0.47405978 \\ -1.36988181 \end{pmatrix} \begin{pmatrix} 7.0829102 \\ -3.81329871 \\ 3.0481365 \end{pmatrix} \begin{pmatrix} 12.86219784 \\ 3.95173109 \\ 0.97349277 \end{pmatrix} \begin{pmatrix} 16.30109667 \\ -1.10550298 \\ 1.74702909 \end{pmatrix}$$

(c) What is the average reconstruction error if reduce dimension to 2D?

解回來之後把每一筆 data 的誤差相加後，再取平均。

6.064383051974749

5.

a.

Handwritten notes on a notebook page:

- Top left: $= V \Lambda^2 (V \Lambda^2)^T$, $| \Lambda = \Lambda^2 \cdot \Lambda^2$, $\perp V X^T$, \because positive definite
- Middle left: \therefore (a) $A A^T = (A A^T)^T = A \cdot A^T$, $A^T A = (A^T A)^T = A^T A$
- Middle right: Both are symmetric
- Below middle: $x^T A A^T x = (x^T A)(x^T A)^T$, $= \|x^T A\|^2 \geq 0$
- Bottom left: $x^T A^T A x = (x^T A^T)(x^T A^T)^T$, $= \|x^T A^T\|^2 \geq 0$
- Bottom right: Both are semi-positive definite
- Bottom center: (3) They both are symmetric, share same nonzero eigen value

$$3, AA^T x = \lambda x : + \lambda \neq 0$$

for AA^T : ~~$AA^T(Ax) = \lambda(Ax) \neq \|A\|^2 Ax = \lambda Ax$~~

$$\lambda = \|A\|^2$$

~~$\therefore A^T A: A^T A(A^T x) = \lambda(A^T x) +$~~

(C), for AA^T : $AA^T x = \lambda_1 x$.

if $\lambda \neq 0$, ~~$A^T A(A^T x) = \lambda^2 x$~~ $\Rightarrow \lambda_1^2 = \|A\|^2$

for $A^T A$: $A^T A x = \lambda_2 x = \lambda_1 \frac{A^T x}{\|A\|^2} \Rightarrow \lambda_2^2 = \|A\|^2$

if $\lambda \neq 0$, $AA^T Ax = A(\lambda x) = \lambda_2(Ax) \Rightarrow \lambda_2^2 = \|A\|^2$

$\therefore \text{Show } \lambda_1 = \lambda_2$

$\therefore \text{Share non-zero eigen value.}$

b.

$\therefore \text{Show } \lambda_1 = \lambda_2 = \|A\|^2$

$\therefore \text{Share non-zero eigen value.}$

2-1b) $\therefore \Sigma$ is symmetric

$$\Sigma = U \Lambda U^T$$

(U is orthogonal matrix) ($\Lambda = \begin{pmatrix} \sigma_{11} & 0 & \cdots & 0 \\ 0 & \sigma_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{nn} \end{pmatrix}$)

$$\bar{\Sigma} = U \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} U^T = U \Lambda^{\frac{1}{2}} (U \Lambda^{\frac{1}{2}})^T \quad \left| \begin{array}{l} \Lambda = \Lambda^{\frac{1}{2}} \cdot \Lambda^{\frac{1}{2}} \\ \therefore \text{positive definite} \end{array} \right.$$

$$= \frac{1}{n} X X^T \quad (\because \sigma_{ij} \geq 0)$$

$$= \left(\frac{X}{\sqrt{n}} \right) \left(\frac{X^T}{\sqrt{n}} \right) \quad X \in \mathbb{M}_{m \times n} = (x_1 \dots x_n)$$

choose $\underline{x} = \sqrt{n} U \Lambda^{\frac{1}{2}}$

c.

$\mathcal{J}(C) = \text{trace}(\Phi^T \Phi)$

$= \frac{1}{n} + \text{trace}(\Phi^T X^T \Phi)$

$= \frac{1}{n} \sum_{i=1}^n \| \Phi^T X_i \|^2 = \frac{1}{n} \sum_{i=1}^n \| \hat{x}_i^{(s)} \|^2$

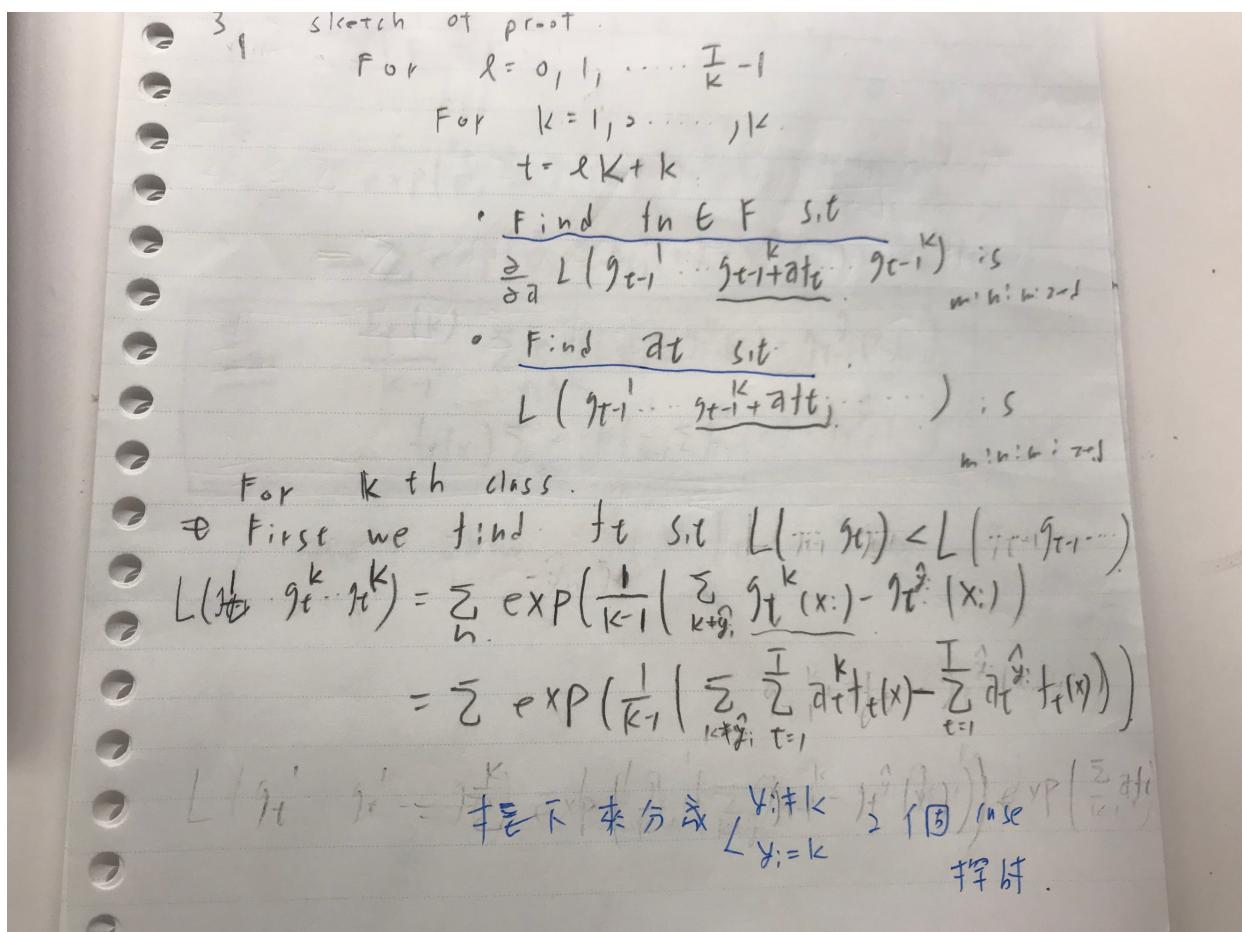
In order to minimize $\| \hat{x}_i^{(s)} \|^2$, so we need to choose the smallest eigen value of $\Phi^T \Phi$.
And PCA takes the largest k eigenvectors to minimize $\| \hat{x}_i^{(s)} \|^2$.

$\min \| \hat{x}_i^{(s)} \|^2$ happens when $\Phi_{OPT} = [u_1, u_2, \dots, u_k]$

By: $\text{Span}\{u_1, \dots, u_m\} = \text{Span}\{x_1, \dots, x_n\}$

$\Phi_{PCA} = \sum_{i=1}^k u_i$

6.



$$\begin{aligned}
& \quad \text{Left side: } A^T A = (A A^T)^T = A \cdot A^T \\
& = \sum_{y_i \neq k}^n \exp \left(\frac{1}{k-1} \sum_{k \neq y_i}^k g_{T-1}^k(x_i) + \frac{\hat{a}_t^k f_t(x)}{k-1} - g_T^y(x_i) \right) + \\
& \quad \sum_{y_i=k} \exp \left(\frac{1}{k-1} \sum_{k \neq y_i}^k g_{T-1}^k(x_i) - \frac{\hat{a}_t^k f_t(x)}{k-1} - g_T^y(x_i) \right) \\
& \Delta L = L(g_T^k) - L(g_{T-1}) \stackrel{\approx}{=} + \frac{\hat{a}_t^k f_t(x)}{k-1} \quad (\because \hat{a}_t^k \ll 1) \\
& = \sum_{y_i \neq k} \exp \left(\frac{1}{k-1} \sum_{k \neq y_i}^k g_{T-1}^k(x_i) - g_{T-1}^y(x_i) \right) \left(\exp \left(\frac{\hat{a}_t^k f_t(x)}{k-1} \right) - 1 \right) + \\
& \quad \sum_{y_i=k} \exp \left(\frac{1}{k-1} \sum_{k \neq y_i}^k g_{T-1}^k(x_i) - g_{T-1}^y(x_i) \right) \left(\exp \left(-\hat{a}_t^k f_t(x_i) - 1 \right) \right. \\
& \frac{\partial L}{\partial \hat{a}_t^k} = \frac{1}{k-1} \sum_{y_i \neq k} \exp \left(\frac{1}{k-1} \sum_{k \neq y_i}^k g_{T-1}^k(x_i) - g_{T-1}^y(x_i) \right) - \\
& \Delta L = \sum_{y_i=k} \exp \left(\frac{1}{k-1} \sum_{k \neq y_i}^k g_{T-1}^k(x_i) - g_{T-1}^y(x_i) \right) \frac{\hat{a}_t^k f_t(x)}{k-1} \\
& = \sum \exp \left(\frac{1}{k-1} \sum_{k \neq y_i}^k g_{T-1}^k(x_i) - g_{T-1}^y(x_i) \right) \hat{a}_t^k f_t(x_i) \\
& \boxed{\frac{\partial L}{\partial \hat{a}_t^k} = \frac{f_t(k)}{k-1} \sum_{y_i \neq k} \exp \left(\frac{1}{k-1} \sum_{k \neq y_i}^k g_{T-1}^k(x_i) - g_{T-1}^y(x_i) \right) - f_t(x_i) \sum_{y_i=k} \exp \left(\frac{1}{k-1} \sum_{k \neq y_i}^k g_{T-1}^k(x_i) - g_{T-1}^y(x_i) \right)} \quad Q.E.D. \\
& \text{We want to show } f_t \leftarrow f_{T-1} - h \frac{\partial L}{\partial \hat{a}_t^k} \quad \text{like gradient descent}
\end{aligned}$$

find \bar{a}_t

$$\frac{\partial L}{\partial \bar{a}_t} = \underbrace{\sum_{y_i \neq k} u_i^n e^{\bar{a}_t}}_{z_t \epsilon t} \cdot \frac{1}{k-1} - \underbrace{\sum_{y_i = k} u_i^n e^{-\bar{a}_t}}_{z_t(1-\epsilon t)} = 0$$

$$\Rightarrow z_t \epsilon t e^{\bar{a}_t} \frac{1}{k-1} - z_t (1-\epsilon t) e^{-\bar{a}_t} = 0$$

$$e^{2\bar{a}_t} = \left(\frac{1-\epsilon t}{\epsilon t}\right)^{k-1}$$

$$\bar{a}_t = \ln \overline{\sqrt{\left(\frac{1-\epsilon t}{\epsilon t}\right)^{k-1}}} \quad \#$$