學號：B06507002 系級： 材料三 姓名：林柏勳

1. (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clustrering不用兩種。）
   兩者後面都標準化後再接 PCA降到32維後用kmeans進行clustering。
   Model 1 (降到1024維)

Layer	Output Shape (batch_size,channel,h,w)
Conv2d(3, 8, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))	-1, 8, 16, 16
Conv2d(8, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))	-1, 16, 8, 8
ConvTranspose2d(16, 8, kernel_size=(2, 2), stride=(2, 2))	-1, 8, 16, 16
ConvTranspose2d(8, 3, kernel_size=(2, 2), stride=(2, 2))	-1, 3, 32, 32
Tanh()	-1, 3, 32, 32

   Model 2 (降到2048維)

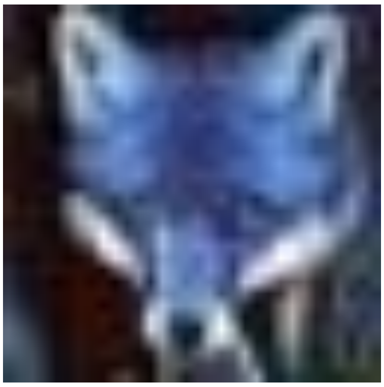
Layer	Output Shape (batch_size,channel,h,w)
Conv2d(3, 8, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))	-1, 8, 16, 16
Conv2d(8, 32 kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))	-1, 32, 8, 8
ConvTranspose2d(32, 8, kernel_size=(2, 2), stride=(2, 2))	-1, 8, 16, 16
ConvTranspose2d(8, 3, kernel_size=(2, 2), stride=(2, 2))	-1, 3, 32, 32
Tanh()	-1, 3, 32, 32

Score or Loss\Model	Model 1	Model 2
Reconstruction Loss	0.04547	0.03782
Public	0.64185	0.63481
Private	0.64968	0.64142

   比較model 1 跟 model 2，雖然model 1 的 reconstruction loss比較大，但是它降的維度比較低，資訊量密度較高，較有利於後續PCA降為後進行clustering，所以分數較高。

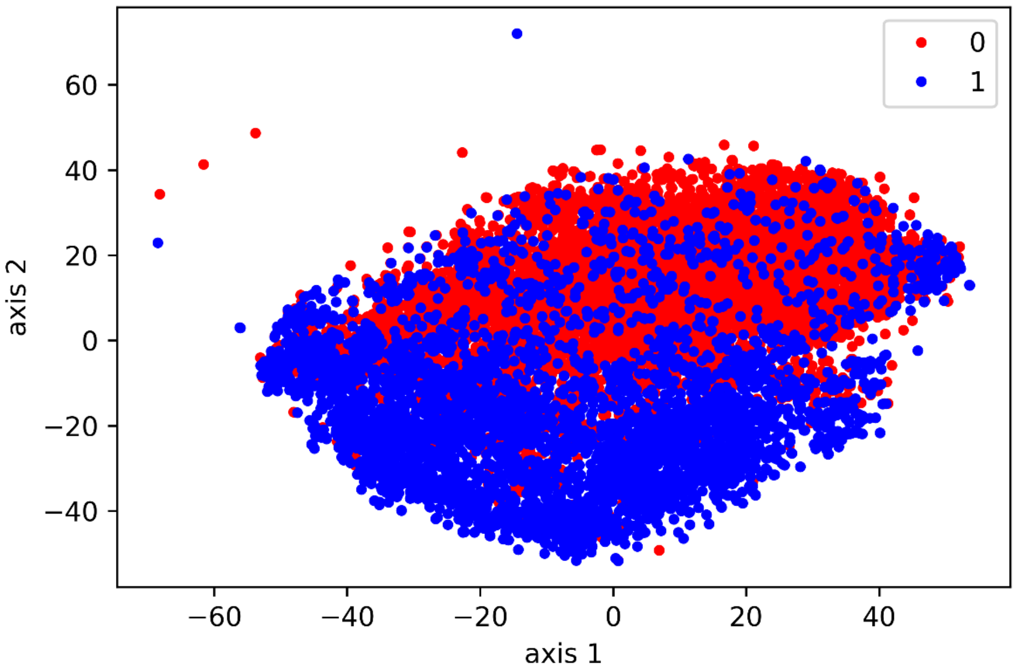
2. (1%) 從dataset選出2張圖，並貼上原圖以及經過autoencoder後reconstruct的圖片。


原圖　　　　　　　　　　reconstruct 後



原圖　　　　　　　　　　reconstruct 後

3. (1%) 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。
   僅管accuracy位於83%左右，但是兩個clustering還是分不太開。



4. (3%)Refer to math problem
   https://drive.google.com/file/d/1e_IDAV2yv0YEhIuVWpDdaH4Pzz5s1p2P/view?fbclid=IwAR0tO9NRxK9JZeUDNdawNuSbGTvqI7niuMX3Kkk9arauC8O6p6iJc7oMz84

1.

(a)

引用套件，將資料轉換為平均值為0的數據，並計算共變異數矩陣。

```
from numpy import linalg as LA
import numpy as np
x = np.array([[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]])
# Calculate the covariance matrix
t=np.mean(x,0)
n,p=np.shape(x)
X=x-t
cov=np.matmul(X.T,X)/(n-1)
print(cov)
```

```
[[13.37777778  0.55555556  3.64444444]
 [ 0.55555556 13.55555556  3.22222222]
 [ 3.64444444  3.22222222  9.06666667]]
```

計算principal axes，即為共變異數矩陣中的eigenvectors。

```
w, v = LA.eig(cov)
w = w[::-1]
for i in range(p):
    v[i,:] = v[i,:][::-1]
for i in range(3):
    print("principal axes %d = "%(i+1),v[i,:], "( eigenvalue=" ,w[i],")")
```

```
principal axes 1 =  [-0.6165947  -0.67817891  0.39985541] ( eigenvalue= 16.99715933101307 )
principal axes 2 =  [-0.58881629  0.73439013  0.33758926] ( eigenvalue= 12.922804099373769 )
principal axes 3 =  [-0.52259579 -0.02728563 -0.85214385] ( eigenvalue= 6.080036569613188 )
```

計算各Sample的Principal Component

```
for i,ele in enumerate(np.dot(X,v)):
    print("Sample %d: Principal Component= "%(i+1),ele)
```

```
Sample 1: Principal Component=  [ 7.18658682 -1.37323947 -2.25104047]
Sample 2: Principal Component=  [ 0.75871342  0.94399334 -0.73022635]
Sample 3: Principal Component=  [-3.07034019  4.45059025 -3.1883001 ]
Sample 4: Principal Component=  [ 2.60849751  2.97853006 -1.92979259]
Sample 5: Principal Component=  [-1.82299166  4.75401212  4.25159619]
Sample 6: Principal Component=  [ 3.35457763 -3.91896138  2.52755823]
Sample 7: Principal Component=  [-4.41464321 -2.55604371 -2.13952468]
Sample 8: Principal Component=  [3.46569126 1.73131477 2.27849363]
Sample 9: Principal Component=  [-2.31359638 -6.03371503  0.2038499 ]
Sample 10: Principal Component=  [-5.75249521 -0.97648096  0.97738622]
```

選取eigenvalue前兩大的eigenvector，對其進行投影，並計算平均reconstruction error

```
v = v[:,:2]
recon=np.dot(np.dot(X,v),v.T)+t
print("average reconstruction error= ",np.sum((recon-x)**2)/n)
```

```
average reconstruction error=  5.47203291265186
```

2.

reference:

https://math.stackexchange.com/questions/3250254/can-xtx-have-negative-eigenvalues

https://statisticaloddsandends.wordpress.com/2018/01/31/xtx-is-always-positive-semidefinite/

http://www.cs.umd.edu/class/summer2019/cmsc320/files/Lec31_July12_2019.pdf

date . . No.

2. (a) $A \in \mathbb{R}^{m \times n}$

proof : AA^T is symmetric

$$(AA^T)^T = (A^T)^T A^T = AA^T$$

proof : A^TA is symmetric

$$(A^TA)^T = A^T (A^T)^T = A^TA$$

proof : AA^T is positive semi-definite

$\forall \ z \in \mathbb{R}^n$

$$z^T(AA^T)z = (A^Tz)^T(A^Tz) = \|A^Tz\|_2^2 \geq 0$$

proof : A^TA is positive semi-definite

$$z^T(A^TA)z = (Az)^T(Az) = \|Az\|_2^2 \geq 0$$

proof : AA^T and A^TA share the same non-zero eigenvalues

multiply by A

$$(A^TA)\phi = \lambda\phi \qquad \phi \in \mathbb{R}^n$$

$$AA^TA\phi = \lambda A\phi \qquad \lambda \text{ can be any non-zero eigenvalues of } A^TA$$

$$AA^T(A\phi) = \lambda(A\phi)$$

$A\phi$: eigenvector of AA^T

2 (b) given data $W^{n \times m}$ [n : number of data, m : number of feature

We can shift all columns of data to zero mean (mean center), which is our data points $X \in \mathbb{R}^{n \times m}$

$$X = W - \begin{bmatrix} \mu_W \\ \vdots \\ \mu_W \end{bmatrix}_{n \times m}, \text{ where } \mu_W = \frac{1}{n}\sum_{i=1}^{n} W_i$$

W_i = i-th row of $W \in \mathbb{R}^m$

$\{X_1, X_2 \cdots X_i \cdots X_n\}$ are our set of data points

X_i : i-th row of $X \in \mathbb{R}^m$

date No.

$$\frac{1}{n}\sum_{i=1}^{n} X_i = 0_m = \mu \in \mathbb{R}^m$$

$$\frac{1}{n}\sum_{i=1}^{n} X_i X_i^T = \frac{1}{n} X^T X = \Sigma$$

$$\Sigma = Z^T\left(\frac{1}{n}X^T X\right) Z = \frac{1}{n}(XZ)^T(XZ) = \frac{1}{n}\|XZ\|_2^2 \geq 0$$

$\therefore$ Give data W, we can construct $\{X_1, \dots X_n\} \in \mathbb{R}^m$ s.t. $\Sigma \in \mathbb{R}^{m\times m}$ be positive semi-definite
$\mu \in \mathbb{R}^m$
and Σ, μ are calculated by the give formula

2-(c) $1 \leq k < m$

minimize $\mathrm{Trace}(\Phi^T \Sigma \Phi)$
subject to $\Phi^T \Phi = I_m$
variables $\Phi \in \mathbb{R}^{m\times k}$

$$\mathrm{Trace}(\Phi^T \Sigma \Phi) = \frac{1}{m}\mathrm{Trace}(\Phi^T X X^T \Phi)$$
$$= \frac{1}{m}\|\Phi^T X\|_F^2 = \frac{1}{m}\sum_{i=1}^{m}\|\Phi^T X_i\|^2$$

$$\Phi = [u_1\ u_2\ u_i\ u_k]$$

where u_1 is the eigenvector corresponding to the smallest eigenvalue of Σ

$\vdots$

u_i is the eigenvector corresponding to the i-th small eigenvalue of Σ

pf: let $\alpha \in \mathbb{R}^k$
minimize $\alpha^T \Sigma \alpha$ subject to $\alpha^T\alpha = 1$
Lagrange multipler
$$\alpha^T\Sigma\alpha - \lambda(\alpha^T\alpha - 1)$$

$$\frac{d}{d\alpha}\left(\alpha^T \Sigma \alpha - \lambda(\alpha^T\alpha - 1)\right) = 0$$

$$\Sigma\alpha - \lambda\alpha = 0 \qquad \Sigma\alpha = \lambda\alpha$$

$$\Rightarrow \quad \alpha^T\Sigma\alpha = \alpha^T\lambda\alpha = \lambda$$

$\because$ the constraint of $\Phi^T\Phi = I_m$

$\therefore$ We have to choose k eigenvectors corresponding to k smallest eigenvalues to ensure they are orthonormal.

$\Phi = [u_1 \dots u_k]$

$$\Phi^T\Sigma\Phi = \begin{bmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_k^T \end{bmatrix} \Sigma \, [u_1 \dots u_k]$$

$$= \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_k \end{bmatrix}$$

$$\text{Trace}(\Phi^T\Sigma\Phi) = \lambda_1 + \dots \lambda_k$$

Note that for $\alpha \in \mathbb{R}^{m \times k}$ $m > 1$, we can set a constraint that the new vectors at row $= m$ is independent from the previous $m-1$ eigenvectors and solve for the constraint OPT. Then we will get that row m in α is the m-th smallest eigenvector in Σ.

3.

Initialize $g_0^k = 0$ for k=1,⋯K and initialize sample weight $u_0^i = 1$ for i=1,⋯n.

For t=1,⋯T:

Fit the data with sample weight u_t^i for i=1,⋯n and get the classifier f_t

For k=1,⋯K

Do gradient descent to find the coefficients α_t^k

$g_t^k \leftarrow g_{t-1}^k + \alpha_t^k f_t$

For i=1,⋯n

$$u_t^i = u_{t-1}^i \exp\left(\frac{1}{K-1}\sum_k \alpha_{t-1}^k f_{-1t}(\boldsymbol{x_i}) - \frac{K}{K-1}\alpha_{t-1}^{\hat{y}_i} f_{t-1}(\boldsymbol{x_i})\right)$$

At time t, the loss L is given by

$$\mathrm{L} = \sum_{i=1}^{n} \exp(\frac{1}{K-1}\sum_{k\neq\hat{y}_i} g_t^k(\boldsymbol{x_i}) - g_t^{\hat{y}_i}(\boldsymbol{x_i})) = \sum_{i=1}^{n} \exp\left(\frac{1}{K-1}\sum_k g_t^k(\boldsymbol{x_i}) - \frac{K}{K-1} g_t^{\hat{y}_i}(\boldsymbol{x_i})\right)$$

$$= \sum_{i=1}^{n} u_t^i \exp\left(\frac{1}{K-1}\sum_k \alpha_t^k f_t(\boldsymbol{x_i}) - \frac{K}{K-1}\alpha_t^{\hat{y}_i} f_t(\boldsymbol{x_i})\right)$$

where

$$u_t^i = \exp\left(\frac{1}{K-1}\sum_k g_{t-1}^k(\boldsymbol{x_i}) - \frac{K}{K-1} g_{t-1}^{\hat{y}_i}(\boldsymbol{x_i})\right)$$

$$u_{t+1}^i = u_t^i \exp\left(\frac{1}{K-1}\sum_k \alpha_t^k f_t(\boldsymbol{x_i}) - \frac{K}{K-1}\alpha_t^{\hat{y}_i} f_t(\boldsymbol{x_i})\right)$$

To get the coefficient α_t^k , we must calculate

$$\frac{\partial L}{\partial \alpha_t^k} = \sum_{i=1}^{n} \exp\left(\frac{1}{K-1}\sum_k (g_{t-1}^k(\boldsymbol{x_i}) + \alpha_t^k f_t(\boldsymbol{x_i})) - \frac{K}{K-1}(g_{t-1}^{\hat{y}_i}(\boldsymbol{x_i}) + \alpha_t^{\hat{y}_i} f_t(\boldsymbol{x_i}))\right)$$

$$\times\left(\frac{1}{K-1} - \frac{K}{K-1}\delta(k-\hat{y}_i)\right) f_t(\boldsymbol{x_i}) = 0$$

$$=> \sum_{k\neq\hat{y}} \exp\left(\frac{1}{K-1}\sum_k (g_{t-1}^k(\boldsymbol{x_i}) + \alpha_t^k f_t(\boldsymbol{x_i}))\right) + \sum_{k=\hat{y}} \exp\left(-(g_{t-1}^{\hat{y}_i}(\boldsymbol{x_i}) + \alpha_t^{\hat{y}_i} f_t(\boldsymbol{x_i}))\right) = 0$$

Since this equation has no explicit solution, we can solve it by numerical method to get α_k^t.