機器學習概論作業

範圍： Autoencoder

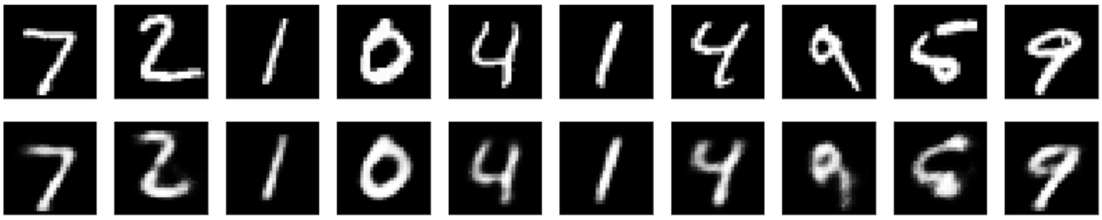
銘傳大學電腦與通訊工程系

班　級	電通三乙
姓　名	李柏賢
學　號	07050862
作業成果	應繳作業共 2 題，第一題 80 分，第二題 20 分。 我共完成 2 題，應得 95 分
授課教師	陳慶逸

75.425+19.761=95.186

■ 請確實填寫自己寫完成題數，填寫不實者(如上傳與作業明顯無關的答案，或是計算題數有誤者)，本次作業先扣 50 分。

一、 建立一個 Autoencoder 架構，中間的瓶頸層為 32；產生 val_loss 和下面的圖片重建結果 (本題佔 80 分，分數計算原則 80 – (val_loss*50)):



程式碼:

```
import keras
from keras import layers

# This is the size of our encoded representations
encoding_dim = 32  # 32 floats -> compression of factor 24.5, assuming the input is 784 floats

# This is our input image
input_img = keras.Input(shape=(784,))
# "encoded" is the encoded representation of the input
encoded = layers.Dense(encoding_dim, activation='relu')(input_img)
# "decoded" is the lossy reconstruction of the input
decoded = layers.Dense(784, activation='sigmoid')(encoded)

# This model maps an input to its reconstruction
autoencoder = keras.Model(input_img, decoded)
# This model maps an input to its encoded representation
encoder = keras.Model(input_img, encoded)
# This is our encoded (32-dimensional) input
encoded_input = keras.Input(shape=(encoding_dim,))
# Retrieve the last layer of the autoencoder model
decoder_layer = autoencoder.layers[-1]
# Create the decoder model
decoder = keras.Model(encoded_input, decoder_layer(encoded_input))
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

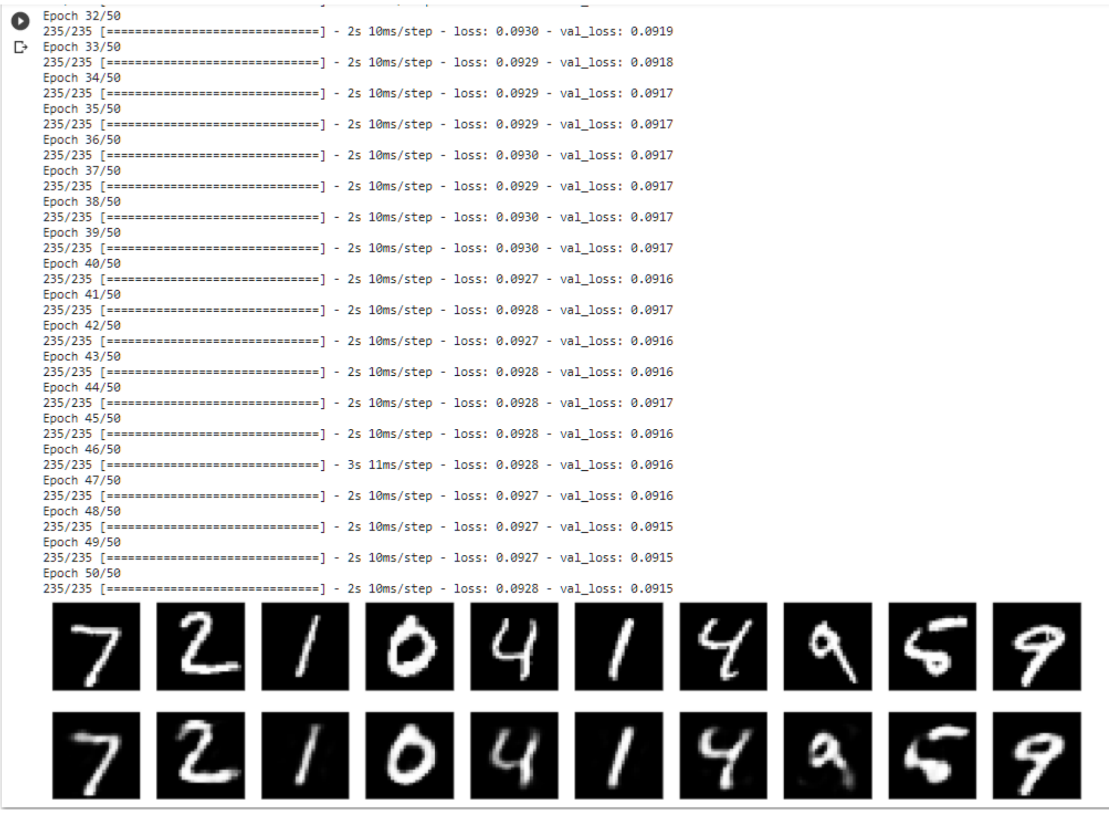
```
from keras.datasets import mnist
import numpy as np
(x_train, _), (x_test, _) = mnist.load_data()
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1
:])))
x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:]))
)
print(x_train.shape)
print(x_test.shape)
autoencoder.fit(x_train, x_train,
                epochs=50,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))
# Encode and decode some digits
# Note that we take them from the *test* set
encoded_imgs = encoder.predict(x_test)
decoded_imgs = decoder.predict(encoded_imgs)

import matplotlib.pyplot as plt

n = 10 # 我們想展示圖像的數量
plt.figure(figsize=(20, 4))
for i in range(n):
    # 秀出原圖像
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    # 秀出重建圖像
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```

```
plt.show()
```

輸出結果擷圖:



```
Epoch 32/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0930 - val_loss: 0.0919
Epoch 33/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0929 - val_loss: 0.0918
Epoch 34/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0929 - val_loss: 0.0917
Epoch 35/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0929 - val_loss: 0.0917
Epoch 36/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0930 - val_loss: 0.0917
Epoch 37/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0929 - val_loss: 0.0917
Epoch 38/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0930 - val_loss: 0.0917
Epoch 39/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0930 - val_loss: 0.0917
Epoch 40/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0927 - val_loss: 0.0916
Epoch 41/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0928 - val_loss: 0.0917
Epoch 42/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0927 - val_loss: 0.0916
Epoch 43/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0928 - val_loss: 0.0916
Epoch 44/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0928 - val_loss: 0.0917
Epoch 45/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0928 - val_loss: 0.0916
Epoch 46/50
235/235 [==============================] - 3s 11ms/step - loss: 0.0928 - val_loss: 0.0916
Epoch 47/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0927 - val_loss: 0.0916
Epoch 48/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0927 - val_loss: 0.0915
Epoch 49/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0927 - val_loss: 0.0915
Epoch 50/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0928 - val_loss: 0.0915
```

80-(0.0915*50)=75.425

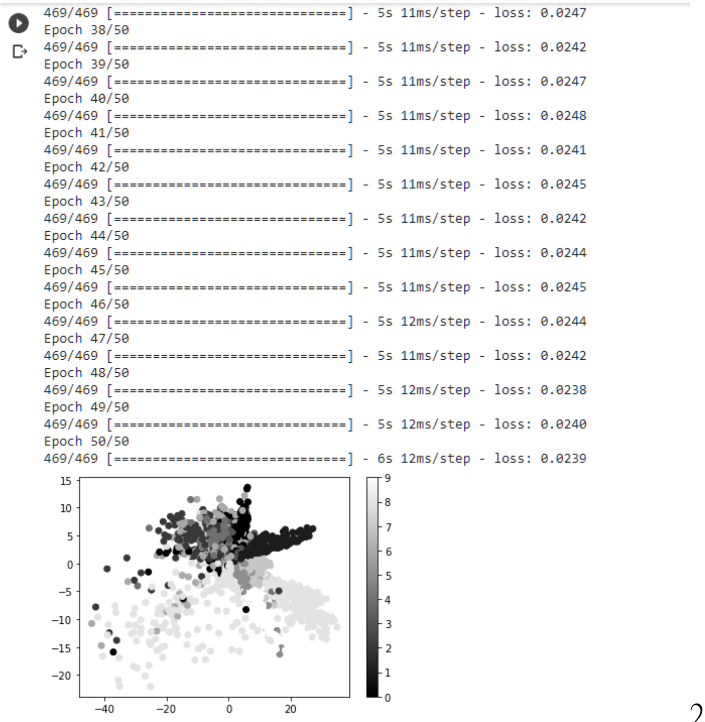
二、　試用一個Autoencoder來學習fashion mnist的資料集(本題佔20分，分數計算原則20 － (val_loss*10)):

程式碼:

```
import numpy as np
np.random.seed(1337) # for reproducibility
from keras.datasets import mnist
from keras.models import Model
from keras.layers import Dense, Input
import matplotlib.pyplot as plt
(x_train, _), (x_test, y_test) = fashion_mnist.load_data()
# data pre-processing
x_train = x_train.astype('float32') / 255. - 0.5 # minmax_normalized
x_test = x_test.astype('float32') / 255. - 0.5 # minmax_normalized
x_train = x_train.reshape((x_train.shape[0], -1))
x_test = x_test.reshape((x_test.shape[0], -1))
print(x_train.shape)
print(x_test.shape)
# in order to plot in a 2D figure
encoding_dim = 2
# this is our input placeholder
input_img = Input(shape=(784,))
# encoder layers
encoded = Dense(256, activation='relu')(input_img)
encoded = Dense(64, activation='relu')(encoded)
encoded = Dense(32, activation='relu')(encoded)
encoded = Dense(10, activation='relu')(encoded)
encoder_output = Dense(encoding_dim)(encoded)
# decoder layers
decoded = Dense(10, activation='relu')(encoder_output)
decoded = Dense(32, activation='relu')(decoded)
decoded = Dense(64, activation='relu')(decoded)
decoded = Dense(256, activation='relu')(decoded)
decoded = Dense(784, activation='tanh')(decoded)
```

```
# construct the autoencoder model
autoencoder = keras.Model(input_img, decoded)
# construct the encoder model for plotting
encoder = keras.Model(input_img, encoder_output)
# compile autoencoder
autoencoder.compile(optimizer='adam', loss='mse')
# training
autoencoder.fit(x_train, x_train,
epochs=50,
batch_size=128,
shuffle=True)
# plotting
encoded_imgs = encoder.predict(x_test)
plt.scatter(encoded_imgs[:, 0], encoded_imgs[:, 1], c=y_test)
plt.colorbar()
plt.show()
```

輸出結果擷圖:



```
469/469 [==============================] - 5s 11ms/step - loss: 0.0247
Epoch 38/50
469/469 [==============================] - 5s 11ms/step - loss: 0.0242
Epoch 39/50
469/469 [==============================] - 5s 11ms/step - loss: 0.0247
Epoch 40/50
469/469 [==============================] - 5s 11ms/step - loss: 0.0248
Epoch 41/50
469/469 [==============================] - 5s 11ms/step - loss: 0.0241
Epoch 42/50
469/469 [==============================] - 5s 11ms/step - loss: 0.0245
Epoch 43/50
469/469 [==============================] - 5s 11ms/step - loss: 0.0242
Epoch 44/50
469/469 [==============================] - 5s 11ms/step - loss: 0.0244
Epoch 45/50
469/469 [==============================] - 5s 11ms/step - loss: 0.0245
Epoch 46/50
469/469 [==============================] - 5s 12ms/step - loss: 0.0244
Epoch 47/50
469/469 [==============================] - 5s 11ms/step - loss: 0.0242
Epoch 48/50
469/469 [==============================] - 5s 12ms/step - loss: 0.0238
Epoch 49/50
469/469 [==============================] - 5s 12ms/step - loss: 0.0240
Epoch 50/50
469/469 [==============================] - 6s 12ms/step - loss: 0.0239
```

20 －(0.0239 *10))=19.761