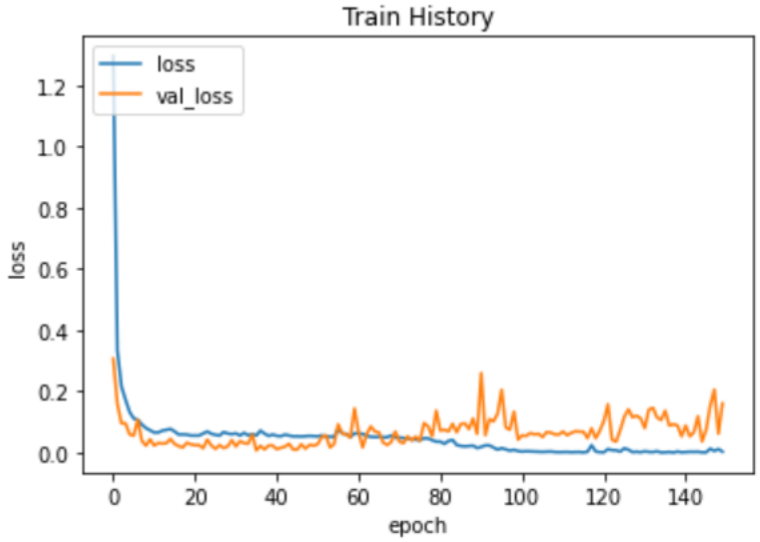
A.

```
Epoch 145/150
69/69 [==============================] - 2s 22ms/step -
y: 0.9918
Epoch 146/150
69/69 [==============================] - 2s 22ms/step -
y: 0.9918
Epoch 147/150
69/69 [==============================] - 1s 21ms/step -
9878
Epoch 148/150
69/69 [==============================] - 1s 21ms/step -
9796
Epoch 149/150
69/69 [==============================] - 1s 22ms/step -
9918
Epoch 150/150
69/69 [==============================] - 1s 21ms/step -
9918
Loss:  0.14521600306034088
Test Accuracy:  0.9764705896377563
```



B.

```
from PIL import Image #用來載入圖檔
import os #設定檔案讀取路徑
from os.path import join #設定檔案讀取路徑
from keras.models import Sequential #建立順序式模型
from keras.layers import Dense, Dropout, Flatten #建立dense layer 與 dropout.layer
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import np_utils #one-hot-encoding
from keras.models import load_model #載入已儲存的模型
import matplotlib.pyplot as plt #畫圖
import numpy as np #處理資料
import random
import keras
from tensorflow.keras.layers import Dropout
```

```
#data_x 與 data_y(label) 資料前處理函式
def data_x_y_preprocess(datapath):
    img_row , img_col =28 ,28 #定義圖片大小
    datapath = datapath  #訓練資料路徑
    data_x=np.zeros((28,28)).reshape(1,28,28) #儲存圖片
    pictureCount =  0 #紀錄照片張數
    data_y=[] #紀錄label
    num_class=10 #數字種類有10種

    #讀取 Image 資料夾內所有檔案
    for root ,dirs ,files in os.walk(datapath):
        for f in files:
            label = int(root.split("\\")[7]) #取得label
            data_y.append(label)#新增標籤給data_y
            fullpath=os.path.join(root , f) #取得檔案路徑
            img = Image.open(fullpath) #開啟img
            img=(np.array(img)/255).reshape(1, 28, 28) #讀取資料時順便做正規劃與reshape
            data_x=np.vstack((data_x,img)) #把array堆疊起來
            pictureCount+=1#用一張照片就+1
    data_x=np.delete(data_x,[0],0) #刪除一開始宣告的np.zeros

    #調整資料格式(圖片張數 , img_row , img_col , 色彩通道 =1 (灰階))
    data_x=data_x.reshape(pictureCount , img_row , img_col , 1)
    data_y=np_utils.to_categorical(data_y , num_class) #將label 轉成 One-Hot Encoding
    return data_x,data_y

(data_train_X,data_train_Y)=data_x_y_preprocess("C:\\Users\\ivan8\\Desktop\\ALG_TERM\\train_image")#呼叫函式並抓取train data
(data_test_X,data_test_Y)=data_x_y_preprocess("C:\\Users\\ivan8\\Desktop\\ALG_TERM\\test_image")#呼叫函式並抓取test data
```

```
#建立模型
model = Sequential() #建立簡單的線性執行的模型
#建立捲積層 filter=32,即 output space的深度, kernel size:3x3, activation function 採用 relu
model.add(Conv2D(32, kernel_size=(3,3), activation='relu' , input_shape=(28,28,1)))
#建立池化層 池化大小= 2x2 取最大值
model.add(MaxPooling2D(pool_size=(2 , 2)))
#建立捲積層 filter=64,即 output size , kernel size:3x3, activation function 採用 relu
model.add(Conv2D(64 , (3,3), activation='relu'))
#建立池化層 池化大小= 2x2 取最大值
model.add(MaxPooling2D(pool_size=(2 , 2)))
#Dropout層隨機斷開輸入神經元，用於防止過度擬合，斷開比例:0.25
model.add(Dropout(0.1))
#Flatten層把多維的輸入一維化，常用在從卷積層到全連結層的過渡
model.add(Flatten())
#Dropout層隨機斷開輸入神經元，用於防止過度擬合，斷開比例:0.25
model.add(Dropout(0.1))
#全連接層 : 128個output
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.25))
#使用 softmax activation function 將結果分類(units=10 表示分10類)
model.add(Dense(units=10 , activation='softmax'))
```

```
#對訓練模型進行設定
model.compile(loss = 'categorical_crossentropy',
              optimizer = 'Adam' ,
              metrics = ['accuracy'])

#進行訓練
train_history = model.fit(data_train_X,data_train_Y , validation_split = 0.1,
                          batch_size = 32 , epochs = 150 , verbose=1)
#進行評估模型準確率
source = model.evaluate(data_test_X,data_test_Y, verbose=0)
print('Loss: ',source[0])
print('Test Accuracy: ',source[1])
```

```
#畫圖顯示預測結果
plt.plot(train_history.history['loss'])
plt.plot(train_history.history['val_loss'])
plt.title('Train History')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['loss','val_loss'],loc='upper left')
plt.show()
#畫出training set 和 validation sat 的 loss
#輸出訓練過程產生的history,訓練資料執行結果,驗證資料執行結果
```