進行資料預處理

1.匯入所需模組

```
from keras.datasets import mnist
from keras.utils import np_utils
import numpy as np
np.random.seed(10)
```

2.讀取mnist資料

```
(x_Train,y_Train),(x_Test,y_Test) = mnist.load_data()
```

```
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
    11493376/11490434 [==============================] - 0s 0us/step
    11501568/11490434 [==============================] - 0s 0us/step
```

3.將features(數字影像特徵值)轉換為4為矩陣

```
x_Train4D=x_Train.reshape(x_Train.shape[0],28,28,1).astype(’float32’)
x_Test4D=x_Test.reshape(x_Test.shape[0],28,28,1).astype(’float32’)
```

4. 將features(數字影像特徵值)標準化

```
x_Train4D_normalize = x_Train4D/255
x_Test4D_normalize = x_Test4D/255
```

5.label(數字的真實地值)以Onehot encoding轉換

```
y_TrainOneHot = np_utils.to_categorical(y_Train)
y_TestOneHot = np_utils.to_categorical(y_Test)
```

建立模型

1.匯入所需模組

```
from keras.models import Sequential
from keras.layers import Dense,Flatten
from keras.layers import Dropout
from keras.layers import Conv2D,MaxPooling2D
```

2. 建立keras的Sequential

```
model  =  Sequential()
```

3. 建立卷積層1

```
model.add(Conv2D(filters=16,                  #建立16個濾鏡filter  weight
                  kernel_size=(5,5),          #每一個濾鏡5*5大小
                  input_shape=(28,28,1),      #第1,2維度:代表輸入的影像形狀28*28大小，第3個維度:因
                  padding='same',            #此設定讓卷積運算，產生的卷積影像大小不變
                  activation='relu'))        #設定Relu激活函數
```

4.建立池化層1

```
model.add(MaxPooling2D(pool_size=(2,2)))
```

5.建立卷積層2

```
model.add(Conv2D(filters=36,            #建立36個濾鏡filter  weight
                kernel_size=(5,5),      #每一個濾鏡filter  weight  5*5大小
                padding='same',          #此設定讓捲積運算並不會改變影像大小
                activation='relu'))      #設定RELU激活函數
```

6.建立池化層2

```
model.add(MaxPooling2D(pool_size=(2,2)))
```

7.加入Dropout避免overfitting

```
model.add(Dropout(0.25))
```

8.建立平坦層

```
model.add(Flatten())
```

9.建立隱藏層，共有128個神經元

```
model.add(Dense(128,activation='relu'))
```

10.加入Dropout層製模型中

```
model.add(Dropout(0.5))
```

11.建立輸出層

```
model.add(Dense(10,activation='softmax'))
```

12.查看模型的摘要

```
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        416       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 36)        14436     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 36)          0         
_________________________________________________________________
dropout (Dropout)            (None, 7, 7, 36)          0         
_________________________________________________________________
flatten (Flatten)            (None, 1764)              0         
_________________________________________________________________
dense (Dense)                (None, 128)               225920    
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290      
=================================================================
Total params: 242,062
Trainable params: 242,062
Non-trainable params: 0
_________________________________________________________________
None
```

進行訓練

1.定義訓練方式

```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

2.開始訓練

```
train_history=model.fit(x=x_Train4D_normalize,y=y_TrainOneHot,validation_split=0.2,epochs=10,batch_
```

```
Epoch 1/10
160/160 - 45s - loss: 0.5193 - accuracy: 0.8377 - val_loss: 0.1016 - val_accuracy: 0.9703
Epoch 2/10
160/160 - 1s - loss: 0.1369 - accuracy: 0.9594 - val_loss: 0.0681 - val_accuracy: 0.9789
```

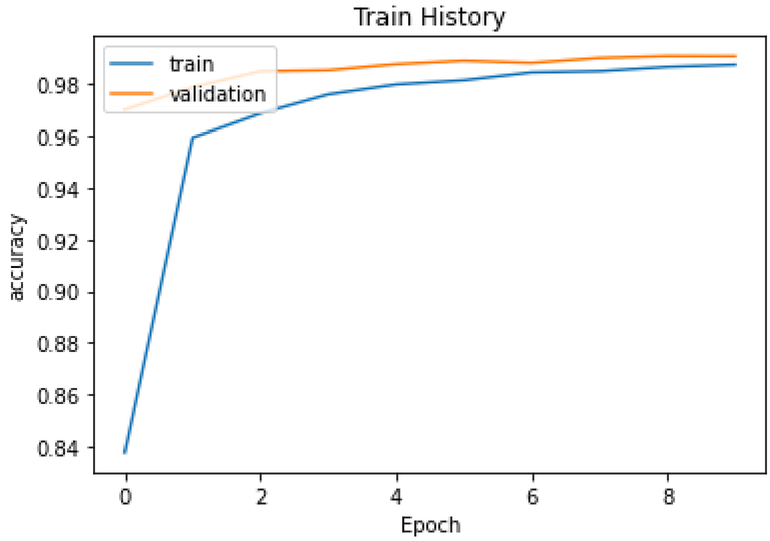
```
Epoch 3/10
160/160 - 1s - loss: 0.1015 - accuracy: 0.9688 - val_loss: 0.0525 - val_accuracy: 0.9851
Epoch 4/10
160/160 - 1s - loss: 0.0784 - accuracy: 0.9761 - val_loss: 0.0482 - val_accuracy: 0.9855
Epoch 5/10
160/160 - 1s - loss: 0.0672 - accuracy: 0.9800 - val_loss: 0.0414 - val_accuracy: 0.9878
Epoch 6/10
160/160 - 1s - loss: 0.0591 - accuracy: 0.9816 - val_loss: 0.0387 - val_accuracy: 0.9892
Epoch 7/10
160/160 - 1s - loss: 0.0515 - accuracy: 0.9846 - val_loss: 0.0417 - val_accuracy: 0.9883
Epoch 8/10
160/160 - 1s - loss: 0.0488 - accuracy: 0.9851 - val_loss: 0.0334 - val_accuracy: 0.9902
Epoch 9/10
160/160 - 1s - loss: 0.0435 - accuracy: 0.9868 - val_loss: 0.0350 - val_accuracy: 0.9910
Epoch 10/10
160/160 - 1s - loss: 0.0401 - accuracy: 0.9876 - val_loss: 0.0326 - val_accuracy: 0.9909
```

3.定義show_train_history函數

```
import  matplotlib.pyplot  as  plt                              #匯入matplotlib.pyplot模組，後續會使用
def  show_train_history(train_history,train,validation):        #定義show_train_history函數，輸入下列
    plt.plot(train_history.history[train])
    plt.plot(train_history.history[validation])
    plt.title(’Train  History’)                                #顯示圖的標題
    plt.ylabel(train)                                          #顯示y軸的標籤
    plt.xlabel(’Epoch’)                                        #設定x軸標籤是’Epoch’
    plt.legend([’train’,’validation’],loc=’upper  left’)        #設定圖例是顯示’train’,’validation’,
    plt.show()
```

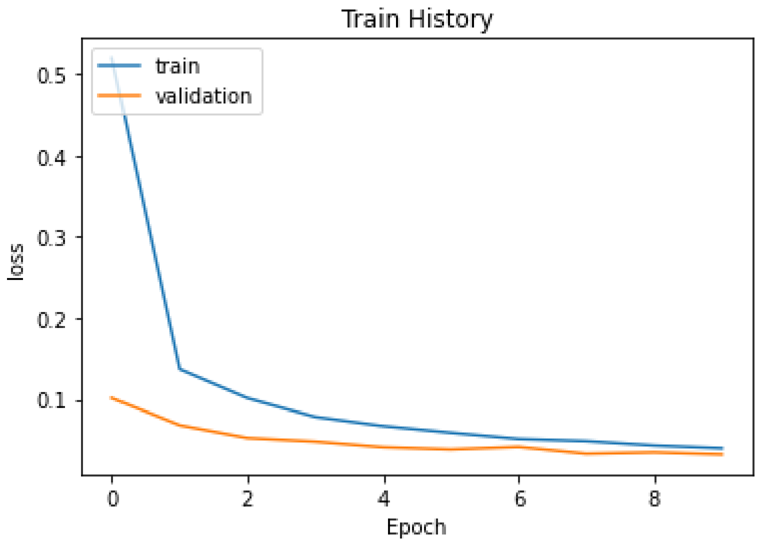
4.畫出accuracy執行結果

```
show_train_history(train_history,’accuracy’,’val_accuracy’)
```



5.畫出loss誤差執行結果

```
show_train_history(train_history,’loss’,’val_loss’)
```



評估模型準確率

1.評估模型準確率

```
scores = model.evaluate(x_Test4D_normalize,y_TestOneHot,verbose=0)
scores[1]
```

```
    0.9927999973297119
```

進行預測

1.執行預測

```
prediction = model.predict_classes(x_Test4D_normalize)
```

```
    /usr/local/lib/python3.7/dist-packages/keras/engine/sequential.py:450: UserWarning: `model.pr
      warnings.warn('`model.predict_classes()` is deprecated and '
```

2.預測結果

```
prediction[:10]
```

```
    array([7, 2, 1, 0, 4, 1, 4, 9, 5, 9])
```

3.建立plot_images_labels_prediction()函數

```
import matplotlib.pyplot as plt    #匯入pyplot模組，後續會使用plt引用
def plot_images_labels_prediction(images,labels,prediction,idx,num=10):    #定義plot_images_labels
    fig = plt.gcf()                                              #設定顯示圖形的大小
    fig.set_size_inches(12,14)                                   #設定顯示圖形的大小
    if num>25:num=25                         #如果顯示筆數參數大於25設定為25，以免發生錯誤
    #for迴圈執行區塊的程式碼，畫出num個數字圖形

    for  i  in  range(0,num):
        ax=plt.subplot(5,5,1+i)      #建立subgraph子圖形為5行5列
        ax.imshow(images[idx],cmap='binary')    #畫出subgraph子圖形
        title  =  'label='+str(labels[idx])      #設定子圖形title，顯示標籤欄位
        if  len(prediction)>0:                  #如果有傳入預測結果
            title+=",predict="+str(prediction[idx])    #標題title加入預測結果
        ax.set_title(title,fontsize=10)             #設定子圖形的標題title大小
        ax.set_xticks([]);ax.set_yticks([])         #設定不顯示刻度
        idx+=1               #讀取下一筆讀取下一筆
        plt.show()      #開始畫圖
```

4.顯示前10筆預測結果

```
plot_images_labels_prediction(x_Test,y_Test,prediction,idx=0)
```

顯示混淆矩陣(confusion matrix)

1.使用pandas crosstab建立混淆矩陣(confusion matrix)

```
import  pandas  as  pd            #匯入pandas模組，後續會以pd引用
pd.crosstab(                      #使用pd.crosstab建立混淆矩陣，輸入下列參數:
        y_Test,              #測試資料數字影像的真實值
        prediction,                #測試資料數字影像的預測結果
        rownames=['labels'],       #設定行的名稱是label
        colnames=['predict'])       #設定列的名稱是predict
```

predict	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
labels										
0	976	0	0	0	0	0	2	1	1	0
1	0	1133	1	0	0	0	1	0	0	0
2	1	0	1029	1	0	0	0	1	0	0
3	0	0	0	1004	0	4	0	0	2	0
4	0	0	0	0	976	0	0	0	1	5
5	1	0	0	5	0	883	2	0	0	1
6	3	2	0	0	1	2	950	0	0	0
7	0	1	6	1	0	0	0	1016	1	3
8	2	0	1	0	0	1	0	1	966	3
9	0	3	0	1	3	3	0	3	1	995