手寫數字辨識應用程式

M11259003 吳柏呈

1. 測試平台及資訊

   平台：Colab

   Tensorflow version : 2.14.0

   資料集：mnist。共 60000 張 28*28 的灰階圖片 (60000,28,28)

   訓練集：50000 張 (50000,28,28)

   測試集：10000 張 (10000,28,28)

2. 模型架構

   從圖一看出共有 7 層 hidden layer，參數分別是 512、256、256、256、128、128、10 以及輸出層的 10，並採用 seLU 作為 activation function。

Layer (type)	Output Shape	Param #
dense_32 (Dense)	(None, 512)	401920
activation_32 (Activation)	(None, 512)	0
dense_33 (Dense)	(None, 256)	131328
activation_33 (Activation)	(None, 256)	0
dense_34 (Dense)	(None, 256)	65792
activation_34 (Activation)	(None, 256)	0
dense_35 (Dense)	(None, 256)	65792
activation_35 (Activation)	(None, 256)	0
dense_36 (Dense)	(None, 128)	32896
activation_36 (Activation)	(None, 128)	0
dense_37 (Dense)	(None, 128)	16512
activation_37 (Activation)	(None, 128)	0
dropout_5 (Dropout)	(None, 128)	0
dense_38 (Dense)	(None, 10)	1290
activation_38 (Activation)	(None, 10)	0

Total params: 715530 (2.73 MB)
Trainable params: 715530 (2.73 MB)
Non-trainable params: 0 (0.00 Byte)

圖 1

3. 超參數及 loss function

```
Optimizer:Adam(lr=0.0001)
Lr = 0.0001
batch_size=64
```

4. 訓練結果分析

圖 2 是訓練次數(X 軸)與準確率(Y 軸)的視覺化，其中藍線代表訓練的準確率變化，可以看到在訓練 5 次後，準確率已經 95%以上，

另外圖 3 代表訓練完後將測試集進行測試，可以得到約 97%的準確。

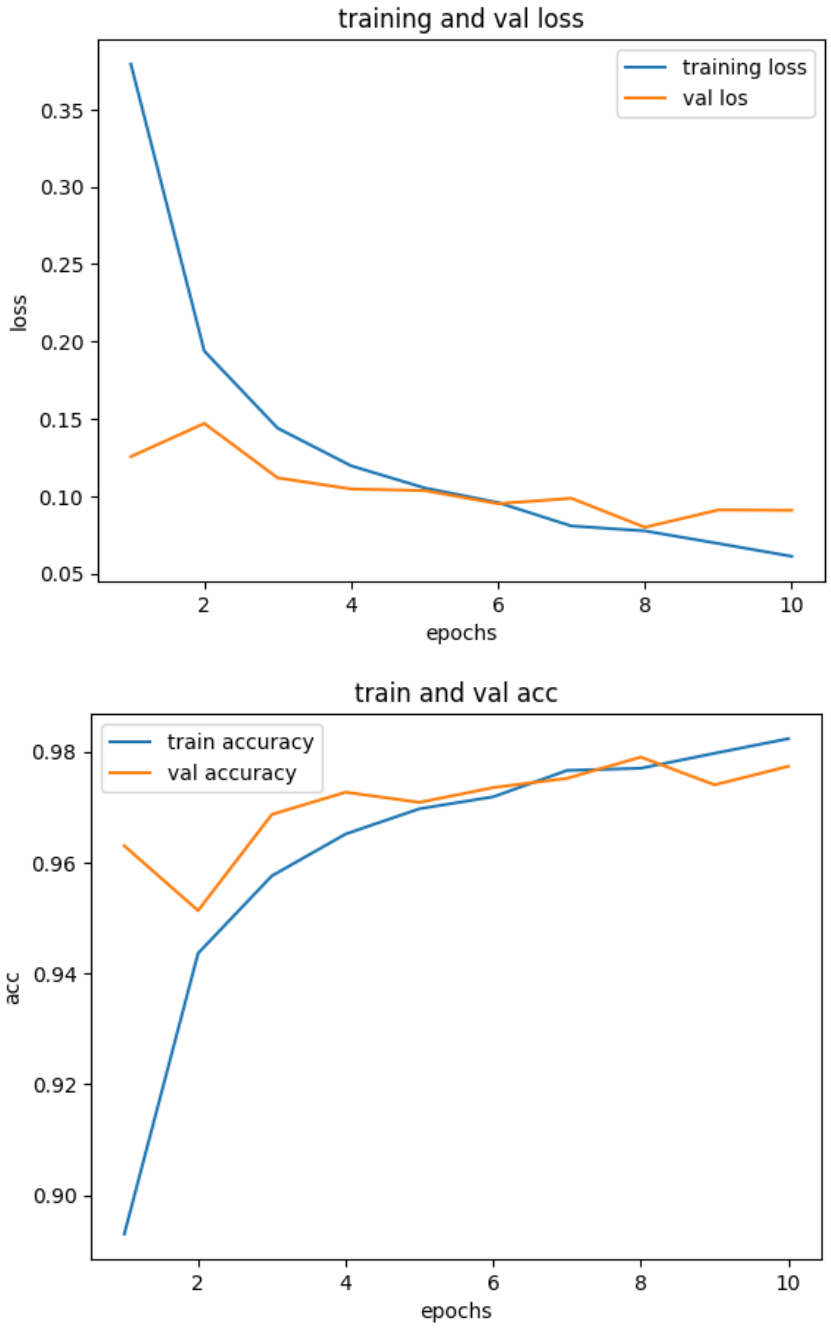


圖 2

```
313/313 [==============================] - 1s 3ms/step
0.9765
```

圖 3

5. Source code

```
import numpy as np
import matplotlib.pyplot as plt
from keras.layers import Dense, Dropout
```

```
from tensorflow.keras.datasets import mnist

(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train=X_train.reshape(len(X_train),-1)/255
X_test=X_test.reshape(len(X_test),-1)/255
# one hot encoding
y_train_one_hot = np.eye(10)[y_train]
y_test_one_hot = np.eye(10)[y_test]

print('Training data shape:',X_train.shape)
print('Testing data shape:',X_test.shape)
```

```
Training data shape: (60000, 784)
Testing data shape: (10000, 784)
```

```
y_train_one_hot[50]
```

```
array([0., 0., 0., 1., 0., 0., 0., 0., 0., 0.])
```

```
y_train[50]
```

```
3
```

```
plt.imshow(X_train[25].reshape(28, 28), cmap='gray')
plt.show()
print(y_train[25])
```

```
import  tensorflow  as  tf
from  tensorflow  import  keras
from  tensorflow.keras.layers  import  Dense,  Activation
from  tensorflow.keras  import  Sequential
from  tensorflow.keras.models  import  Model
from  tensorflow.keras.optimizers  import  Adam

print(tf.__version__)
```

```
2.14.0
```

```
#  此範例使用  Tensorflow2.0  Sequential  API  搭建神經網路。
def  build_model():
        model  =  Sequential()
        model.add(Dense(512,  input_dim=784))
        model.add(Activation('selu'))
        model.add(Dense(256))
        model.add(Activation('selu'))
        model.add(Dense(256))
        model.add(Activation('selu'))
        model.add(Dense(256))
        model.add(Activation('selu'))
        model.add(Dense(128))
        model.add(Activation('selu'))
        model.add(Dense(128))
        model.add(Activation('selu'))

        model.add(Dropout(0.3))
        model.add(Dense(10))
        model.add(Activation('softmax'))
        #model.add(Dense(25,  Activation('relu'),  input_dim=X_train.shape[-1]))
        #model.add(Dense(10,  Activation('softmax')))
        return  model
```

```
model  =  build_model()
model.summary()      #  Weights  =  (784+1)*25+(25+1)*10
```

```
#  編譯模型
optim  =  Adam(lr=0.0001)
model.compile(loss='categorical_crossentropy',
                          optimizer=optim,
                          metrics=['acc'])


#  訓練模型
history  =  model.fit(X_train,  y_train_one_hot,
                                      batch_size=64,
                                      epochs=10,
                                      verbose=1,
                                      shuffle=True,
                                      validation_split=0.1)
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
Epoch 1/10
844/844 [==============================] - 16s 17ms/step - loss: 0.3793 - acc: 0.8930 - val_loss: 0.1257 - val_acc: 0.9630
Epoch 2/10
844/844 [==============================] - 16s 19ms/step - loss: 0.1939 - acc: 0.9436 - val_loss: 0.1471 - val_acc: 0.9513
Epoch 3/10
844/844 [==============================] - 14s 17ms/step - loss: 0.1440 - acc: 0.9576 - val_loss: 0.1119 - val_acc: 0.9687
Epoch 4/10
844/844 [==============================] - 14s 17ms/step - loss: 0.1197 - acc: 0.9651 - val_loss: 0.1048 - val_acc: 0.9727
Epoch 5/10
844/844 [==============================] - 14s 17ms/step - loss: 0.1054 - acc: 0.9697 - val_loss: 0.1037 - val_acc: 0.9708
Epoch 6/10
844/844 [==============================] - 14s 17ms/step - loss: 0.0960 - acc: 0.9718 - val_loss: 0.0952 - val_acc: 0.9735
Epoch 7/10
844/844 [==============================] - 15s 17ms/step - loss: 0.0808 - acc: 0.9766 - val_loss: 0.0987 - val_acc: 0.9752
Epoch 8/10
844/844 [==============================] - 14s 16ms/step - loss: 0.0776 - acc: 0.9770 - val_loss: 0.0800 - val_acc: 0.9790
Epoch 9/10
844/844 [==============================] - 14s 17ms/step - loss: 0.0695 - acc: 0.9797 - val_loss: 0.0912 - val_acc: 0.9740
Epoch 10/10
844/844 [==============================] - 14s 17ms/step - loss: 0.0612 - acc: 0.9823 - val_loss: 0.0910 - val_acc: 0.9773
```

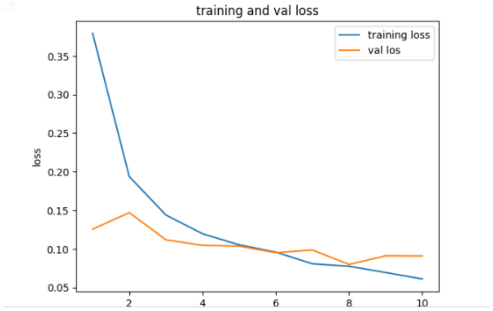
```
history_dict = history.history
history_dict.keys()
```

```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

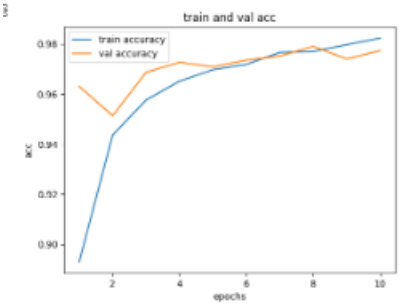
```
import matplotlib.pyplot as plt

acc = history_dict['acc']
val_acc = history_dict['val_acc']
loss = history_dict['loss']
val_loss = history_dict['val_loss']
epochs_ = range(1,len(acc)+1)
```

```
plt.plot(epochs_ , loss , label = 'training loss')
plt.plot(epochs_ , val_loss , label = 'val los')
plt.title('training and val loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```




```
plt.clf()
plt.plot(epochs_ , acc , label='train accuracy')
plt.plot(epochs_ , val_acc , label = 'val accuracy')
plt.title('train and val acc')
plt.xlabel('epochs')
plt.ylabel('acc')
plt.legend()
plt.show()
```




```
from sklearn.metrics import accuracy_score

pred = np.argmax(model.predict(X_test), axis=1)
accuracy_score(y_test, pred)
```

```
313/313 [==============================] - 2s 6ms/step
0.9948
```

```
# now storing some properly as well as misclassified indexes'.
i=0
prop_class=[]
mis_class=[]

for i in range(len(X_test)):
    if(y_test[i]==pred[i]):
        prop_class.append(i)
    if(len(prop_class)==8):
        break

i=0
for i in range(len(X_test)):
    if(y_test[i]!=pred[i]):
        mis_class.append(i)
    if(len(mis_class)==8):
        break
```

```
count=0
fig,ax=plt.subplots(4,2)
fig.set_size_inches(15,15)
for i in range (4):
    for j in range (2):
        ax[i,j].imshow(X_test[prop_class[count]].reshape(28, 28), cmap='gray')
        ax[i,j].set_title("Predicted : "+str(pred[prop_class[count]])+"\n"+"Actual : "+str(y_test[prop_class[count]]))
        plt.tight_layout()
        count+=1
```