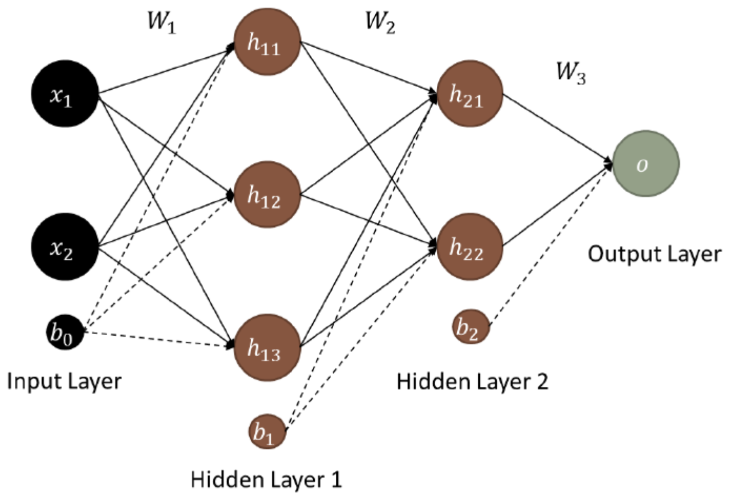
深度學習：基礎與應用_HW3

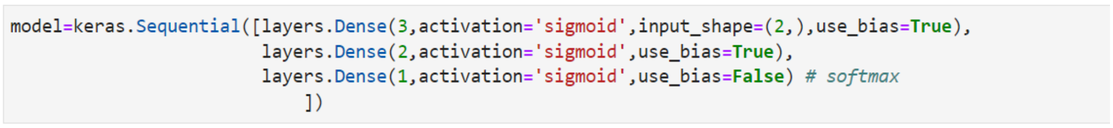
資科碩計一　110753207 林依樺

Q1: (15%) Please use a neural net (with its architecture shown below) to **find the decision boundary** based on 'train.mat.' **The activation function must be used in the two hidden layers and the output layer**. You can use any off-the-shelf functions to construct and optimize your network model. Report the **test error on the test set 'test.mat'** (percentage of misclassified test samples).



- 使用 tensorflow 建立 model

  使用 sigmoid 作為 activation function。



```
model=keras.Sequential([layers.Dense(3,activation='sigmoid',input_shape=(2,),use_bias=True),
                        layers.Dense(2,activation='sigmoid',use_bias=True),
                        layers.Dense(1,activation='sigmoid',use_bias=False) # softmax
                           ])
```

- Model 結構：

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 3)                 9
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 8
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 2
=================================================================
Total params: 19
Trainable params: 19
Non-trainable params: 0
_________________________________________________________________
```

- 使用 Adam 作為 optimizer，loss function 為 BinaryCrossentropy，並用 Accuracy(BinaryAccracy in tensorflow)作為評估指標。

```
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.01),
             loss=keras.losses.BinaryCrossentropy(),
            metrics=[keras.metrics.BinaryAccuracy()]
             )
```

- 訓練模型

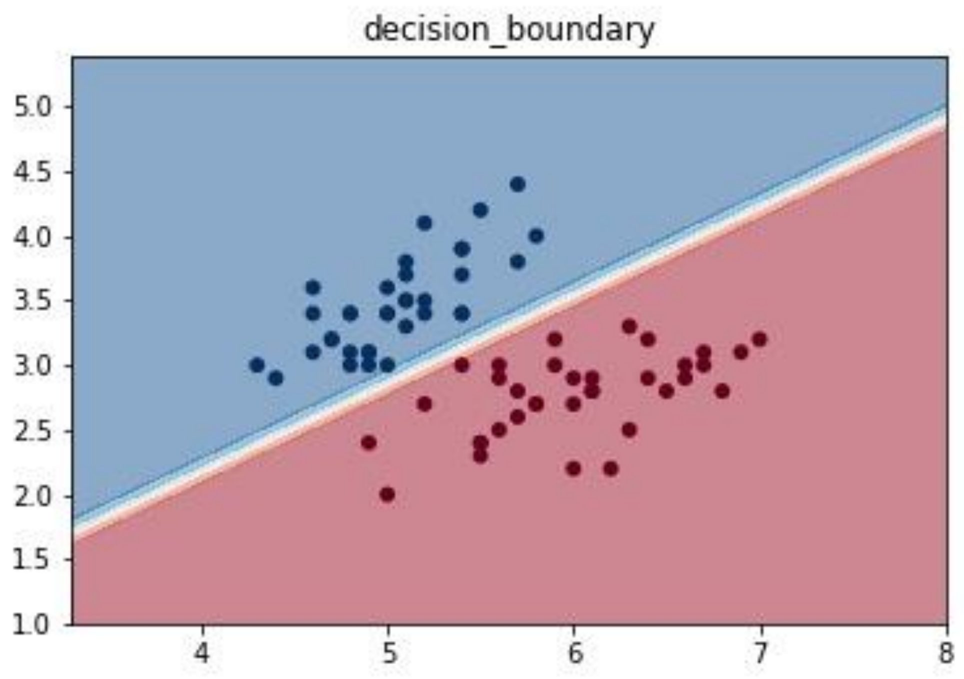
```
history=model.fit(train_x,train_y,batch_size=35,epochs=500
                  # ,callbacks=[model_cbk,model_mckp]
                 )
```

- test error on the test set

```
res = tf.keras.metrics.BinaryAccuracy()
res.update_state(test_y, model.predict(test_x))
print("test error:{}%".format((1-res.result().numpy())*100))
```

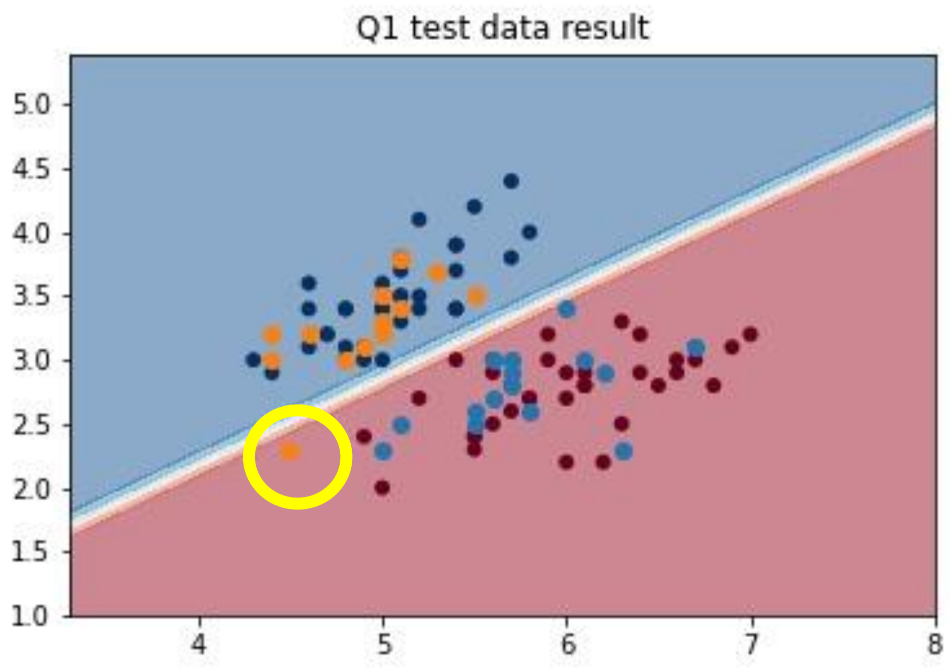
```
test error:3.333336114883423%
```

- decision boundary



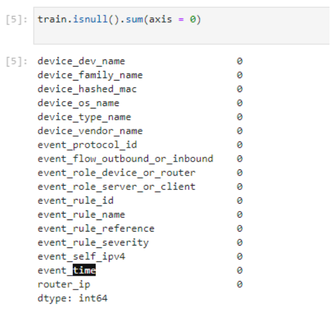
- Test Set 結果

黃色圈起結果為分類錯誤之資料。

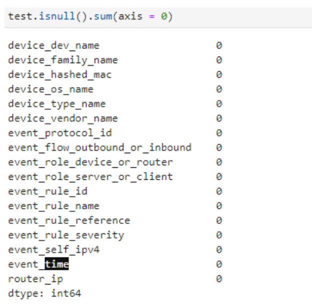


Q2 : Please design a neural net to predict network intrusion types for the datasets “train_DefenseSystem.csv” and “test_DefenseSystem.csv.” Please read the dataset description in the file “DefenseSystem_Readme.docx.”

- 預測目標為 event_rule_category，有兩個分類 Web Attack、Access Control。
- 下載資料集並進行資料前處理。
  - Data 原始的資料型態:
    - 兩個資料集皆無 missing value。

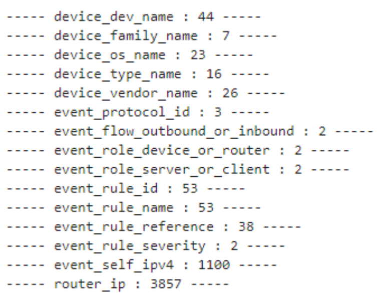


```
[5]: train.isnull().sum(axis = 0)

[5]: device_dev_name                    0
     device_family_name                 0
     device_hashed_mac                  0
     device_os_name                     0
     device_type_name                   0
     device_vendor_name                 0
     event_protocol_id                  0
     event_flow_outbound_or_inbound     0
     event_role_device_or_router        0
     event_role_server_or_client        0
     event_rule_id                      0
     event_rule_name                    0
     event_rule_reference               0
     event_rule_severity                0
     event_self_ipv4                    0
     event_time                         0
     router_ip                          0
     dtype: int64
```



```
test.isnull().sum(axis = 0)

device_dev_name                    0
device_family_name                 0
device_hashed_mac                  0
device_os_name                     0
device_type_name                   0
device_vendor_name                 0
event_protocol_id                  0
event_flow_outbound_or_inbound     0
event_role_device_or_router        0
event_role_server_or_client        0
event_rule_id                      0
event_rule_name                    0
event_rule_reference               0
event_rule_severity                0
event_self_ipv4                    0
event_time                         0
router_ip                          0
dtype: int64
```

    - 刪去欄位為：device_hashed_mac，event_time。
    - Dtype=int64 的 colunm 中，event_protocol_id、enent_rule_id、event_rule_severity 在資料的意涵上屬於類別型態，須轉為 category。其他的 Dtype=object 的 colunm 在資料意涵上也是轉為 category 較佳。
    - 分析欄位有多少不同的 unique value，結果如下:



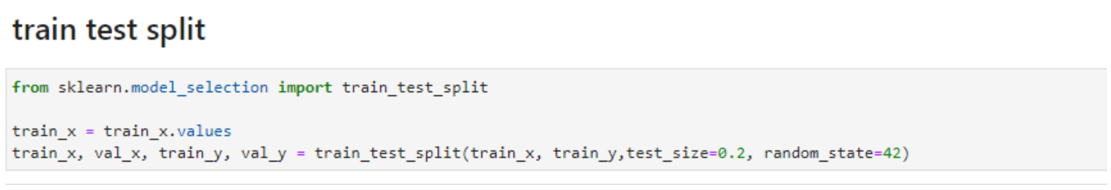
```
----- device_dev_name : 44 -----
----- device_family_name : 7 -----
----- device_os_name : 23 -----
----- device_type_name : 16 -----
----- device_vendor_name : 26 -----
----- event_protocol_id : 3 -----
----- event_flow_outbound_or_inbound : 2 -----
----- event_role_device_or_router : 2 -----
----- event_role_server_or_client : 2 -----
----- event_rule_id : 53 -----
----- event_rule_name : 53 -----
----- event_rule_reference : 38 -----
----- event_rule_severity : 2 -----
----- event_self_ipv4 : 1100 -----
----- router_ip : 3857 -----
```

需要處理 event_self_ipv4、router_ip 非數值資料且 unique value 較多，只保留這個欄位的前三碼為特徵。(def convert2ip_first3 中)

◆ 將類別行欄位進行 one hot emcoding。

2.1. (30%) Randomly select 80% of the training set as your training set and the rest 20% as your validation set. Use the validation set to design your network structure and choose possible hyper-parameters (try 3 different network structures at least and list all the parameters you may use). You need to report the training and validation accuracy for each network.

- 區分訓練集與驗證集，處理過後的資料集共有 524 個特徵。



train test split

```python
from sklearn.model_selection import train_test_split

train_x = train_x.values
train_x, val_x, train_y, val_y = train_test_split(train_x, train_y,test_size=0.2, random_state=42)
```

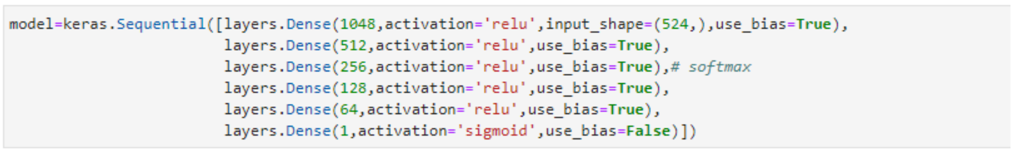
- Model

  使用 Fully connected 的 neural net，調整 neural net,的 layer 數量、寬度以及是否使用 dropout。Epoch 都固定為 150，batch size 為 1000。

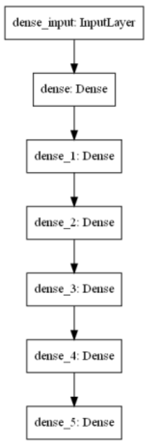
  ■ Model 1

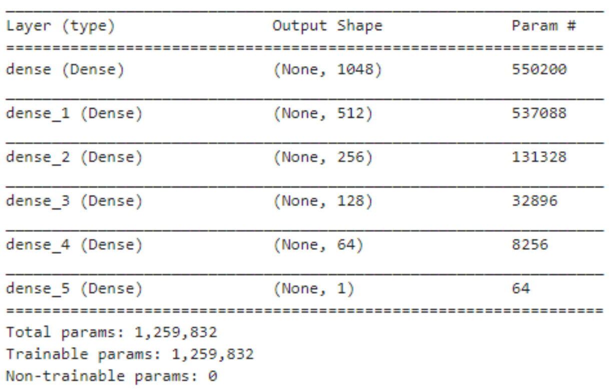
  包含 input layer、output layer，使用 7 層 layer。第一層 hidden layer 的 Shape 為 featrue 特徵的兩倍(512*2=1024)。之後的依序為上一層的 1/2，直到該層的 shape=64 後，每一層 layer 之 activation function 為 relu。透過 sigmoid 輸出結果為 Web Attack 的機率。

  ◆ Model 資訊與訓練程式碼:



```python
model=keras.Sequential([layers.Dense(1048,activation='relu',input_shape=(524,),use_bias=True),
                        layers.Dense(512,activation='relu',use_bias=True),
                        layers.Dense(256,activation='relu',use_bias=True),# softmax
                        layers.Dense(128,activation='relu',use_bias=True),
                        layers.Dense(64,activation='relu',use_bias=True),
                        layers.Dense(1,activation='sigmoid',use_bias=False)])
```



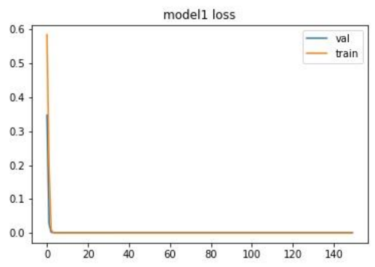


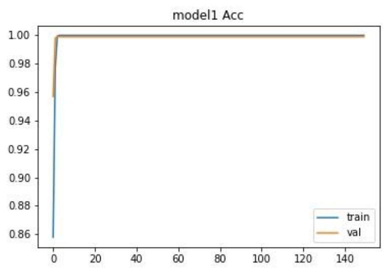
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1048)              550200
_________________________________________________________________
dense_1 (Dense)              (None, 512)               537088
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 64
=================================================================
Total params: 1,259,832
Trainable params: 1,259,832
Non-trainable params: 0
```

```
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001),
                loss=keras.losses.BinaryCrossentropy(),
              metrics=[keras.metrics.BinaryAccuracy()]
                )
```

```
history=model.fit(train_x,train_y,batch_size=1000,epochs=150,
                      validation_data  =  (val_x,val_y),callbacks=[model_cbk,model_mckp]
                  )
```

◆ 訓練結果

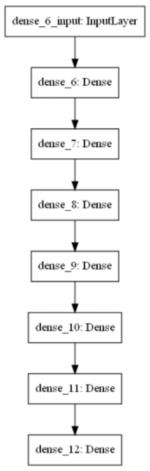




■ Model 2

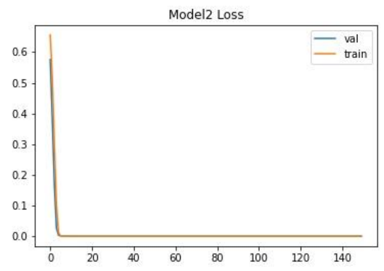
◆ 層數增加，但改變每一層的神經元數量改變為 input layer 為 524，hidden layer 依序為 512、256、128、64、32、16，皆使用 relu，最後 output 經過 sigmoid，輸出的 shape 為 1。

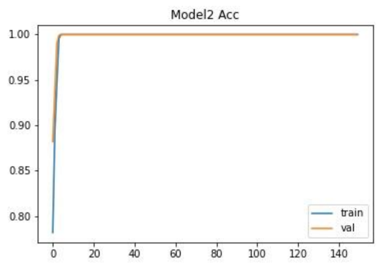
```
model_2=keras.Sequential([layers.Dense(512,activation='relu',input_shape=(524,),use_bias=True),
                          # layers.Dense(512,activation='relu',use_bias=True),
                          layers.Dense(256,activation='relu',use_bias=True),# softmax
                          layers.Dense(128,activation='relu',use_bias=True),
                          layers.Dense(64,activation='relu',use_bias=True),
                          layers.Dense(32,activation='relu',use_bias=True),
                          layers.Dense(16,activation='relu',use_bias=True),
                          layers.Dense(1,activation='sigmoid',use_bias=False)])

model_2.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001),
             loss=keras.losses.BinaryCrossentropy(),
            metrics=[keras.metrics.BinaryAccuracy()]
             )
```



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1048)              550200
_________________________________________________________________
dense_1 (Dense)              (None, 512)               537088
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 64
=================================================================
Total params: 1,259,832
Trainable params: 1,259,832
Non-trainable params: 0
_________________________________________________________________
```

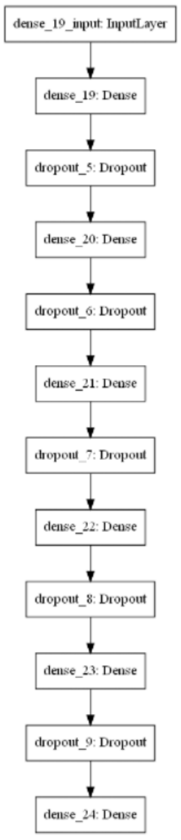
```
model_cbk=keras.callbacks.TensorBoard(log_dir=log_dir)
model_mckp=keras.callbacks.ModelCheckpoint(model_dir+'/Best_model.h5',save_best_only=True)
history_2=model_2.fit(train_x,train_y,batch_size=1000,epochs=150,
                      validation_data  =  (val_x,val_y),callbacks=[model_cbk,model_mckp]
                     )
```

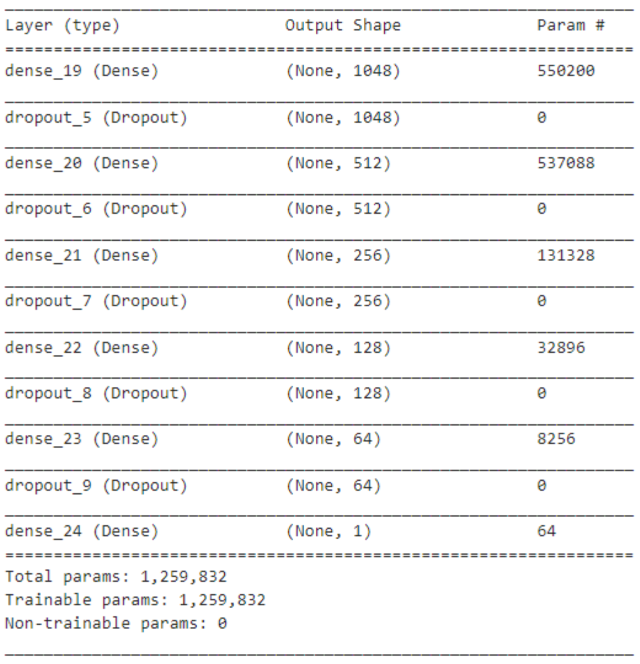




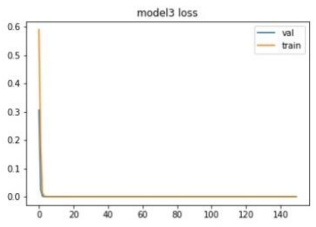
- Model 3
  - 基本維持與 model 1 相同的架構但是每一個隱藏層皆增加 dropout layer(比例為 0.1)

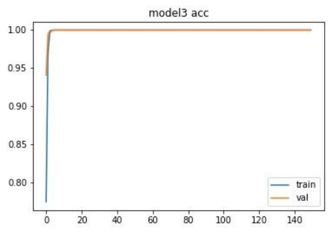
```
model_3=keras.Sequential([layers.Dense(1048,activation='relu',input_shape=(524,),use_bias=True),
                          layers.Dropout(0.1, seed=5),
                          layers.Dense(512,activation='relu',use_bias=True),
                          layers.Dropout(0.1, seed=5),
                          layers.Dense(256,activation='relu',use_bias=True),# softmax
                          layers.Dropout(0.1, seed=5),
                          layers.Dense(128,activation='relu',use_bias=True),
                          layers.Dropout(0.1, seed=5),
                          layers.Dense(64,activation='relu',use_bias=True),
                          layers.Dropout(0.1, seed=5),
                          layers.Dense(1,activation='sigmoid',use_bias=False)])

model_3.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001),
             loss=keras.losses.BinaryCrossentropy(),
            metrics=[keras.metrics.BinaryAccuracy()]
             )
```





```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_19 (Dense)             (None, 1048)              550200
_________________________________________________________________
dropout_5 (Dropout)          (None, 1048)              0
_________________________________________________________________
dense_20 (Dense)             (None, 512)               537088
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_21 (Dense)             (None, 256)               131328
_________________________________________________________________
dropout_7 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_22 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_8 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_23 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_9 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_24 (Dense)             (None, 1)                 64
=================================================================
Total params: 1,259,832
Trainable params: 1,259,832
Non-trainable params: 0
_________________________________________________________________
```



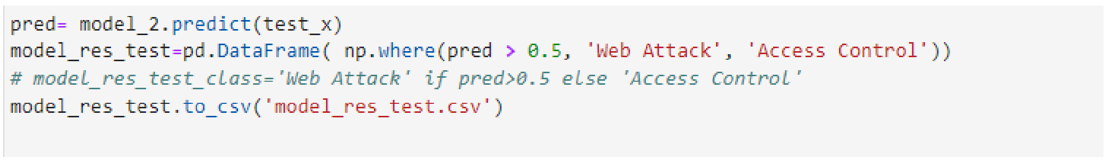


● 最終結果

Model 2 的準確率最高(圖片為四捨五入至小數點第 4 位的結果)。

```
--------------- Accuracy ---------------
                mode1   model2  model3
Train data      1.0     1.0     1.0
Validation      0.9990  1.0000  0.9990
```

2.2. (15%) Use your best network (out of 3 at least) to predict the intrusion type for the test set. Please report your results.



```
pred= model_2.predict(test_x)
model_res_test=pd.DataFrame( np.where(pred > 0.5, 'Web Attack', 'Access Control'))
# model_res_test_class='Web Attack' if pred>0.5 else 'Access Control'
model_res_test.to_csv('model_res_test.csv')
```

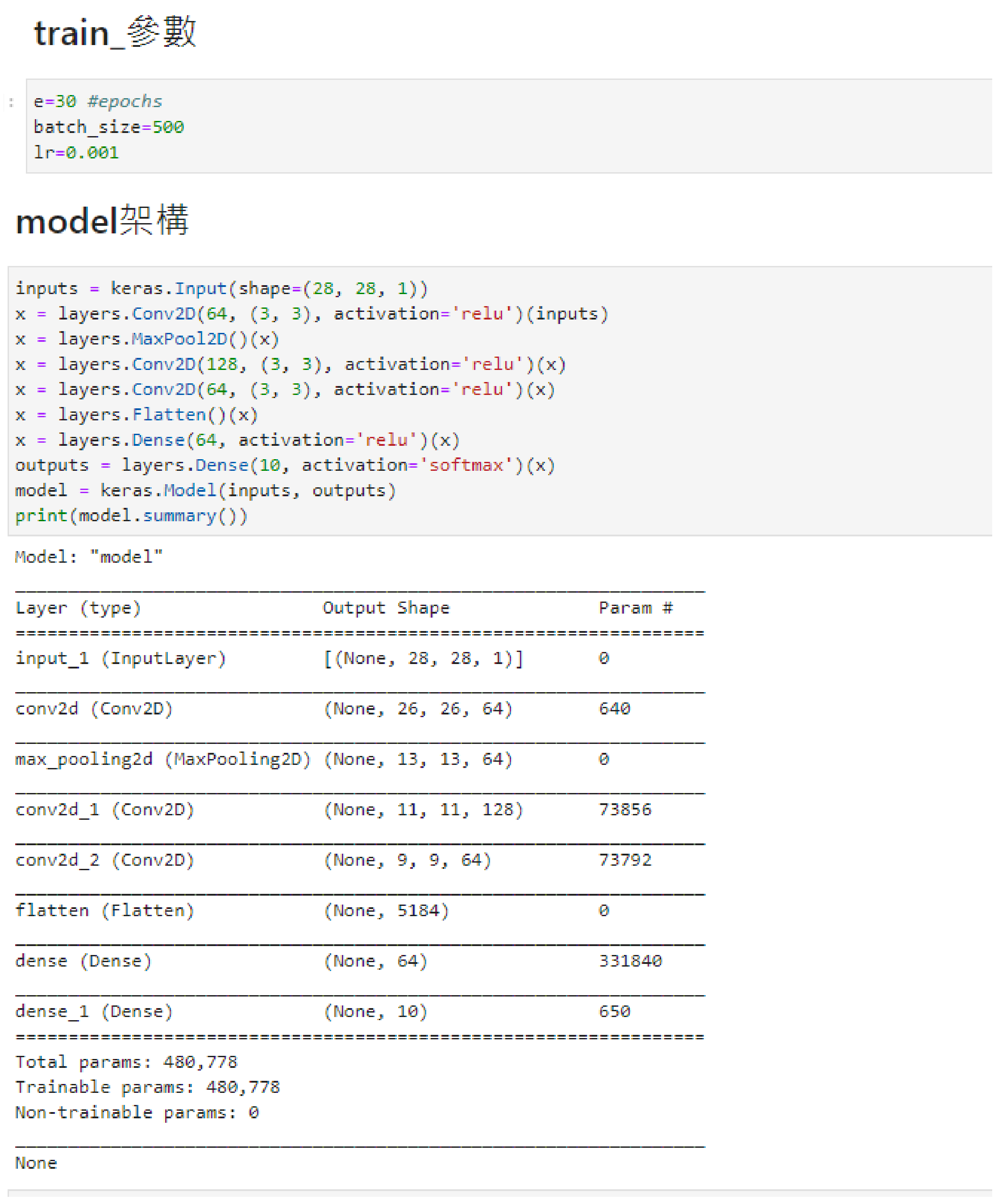
產生 csv 檔案，結果放入繳交檔案中的 Q2 資料夾的 model_res_test.csv 中。

3. (60%) The mnist dataset contains handwritten digits, where it has a training set of 60,000 examples, and a test set of 10,000 examples.

Please download it here: http://yann.lecun.com/exdb/mnist/

3.1 (10%) You are asked to construct a classification model based on convolutional neural networks for digit recognition. Please report the prediction accuracy for the testing set.

- 建立 CNN model 並訓練:



train_參數

```
e=30 #epochs
batch_size=500
lr=0.001
```

model架構

```
inputs = keras.Input(shape=(28, 28, 1))
x = layers.Conv2D(64, (3, 3), activation='relu')(inputs)
x = layers.MaxPool2D()(x)
x = layers.Conv2D(128, (3, 3), activation='relu')(x)
x = layers.Conv2D(64, (3, 3), activation='relu')(x)
x = layers.Flatten()(x)
x = layers.Dense(64, activation='relu')(x)
outputs = layers.Dense(10, activation='softmax')(x)
model = keras.Model(inputs, outputs)
print(model.summary())
```

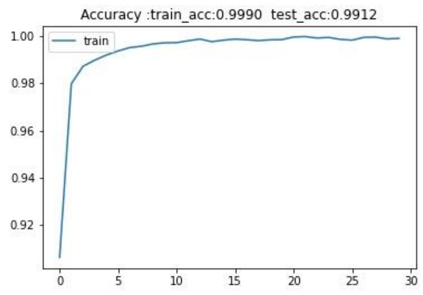
```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 28, 28, 1)]       0         
_________________________________________________________________
conv2d (Conv2D)              (None, 26, 26, 64)        640       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 64)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 128)       73856     
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 9, 9, 64)          73792     
_________________________________________________________________
flatten (Flatten)            (None, 5184)              0         
_________________________________________________________________
dense (Dense)                (None, 64)                331840    
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650       
=================================================================
Total params: 480,778
Trainable params: 480,778
Non-trainable params: 0
_________________________________________________________________
None
```

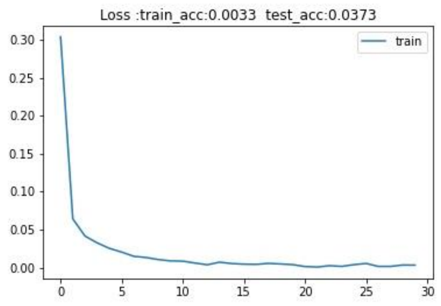
```
model.compile(keras.optimizers.Adam(learning_rate=lr),
              loss=keras.losses.CategoricalCrossentropy(),
              metrics=[keras.metrics.CategoricalAccuracy()])

# 訓練網路模型
CNN_history=model.fit(train_x,train_y,batch_size=batch_size,
             epochs=e
             ,validation_data=(test_x,test_y)
                         # ,
             # callbacks=[model_cbk, model_mckp]
            )
```

- 訓練結果:

  Train 之準確率約為: 0.9990

  Test 之準確率約為:0.9912。

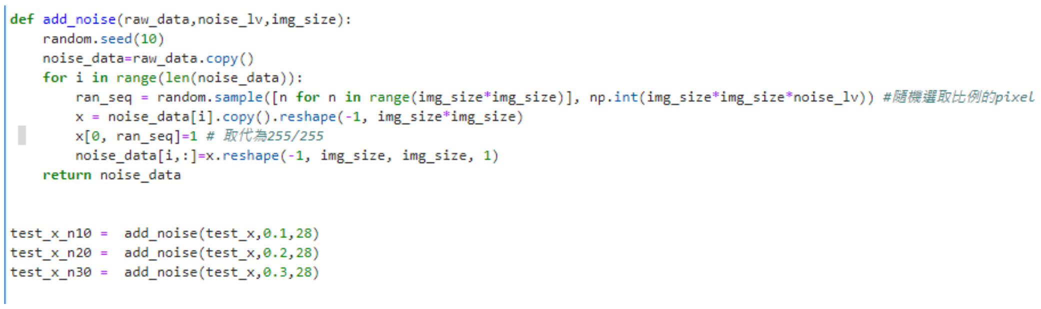




```
: model.evaluate(test_x,test_y)

  313/313 [==============================] - 1s 2ms/step - loss: 0.0373 - categorical_accuracy: 0.9912
: [0.03730926662683487, 0.9911999702453613]
```

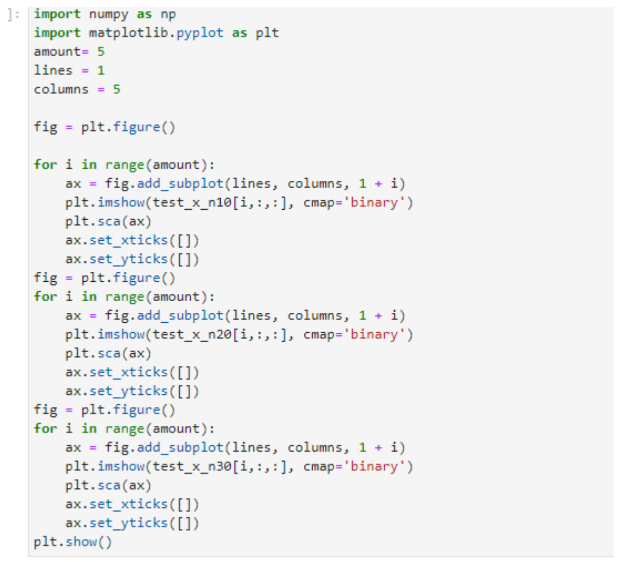
3.2 (30%) Please add 10%, 20%, 30% of salt-and-pepper noise to the test images, and test them using the model you trained on clean images from 3.1. Report the prediction accuracies. Compare your results with those from 3.1. What do you find?

- 生成 nioise:

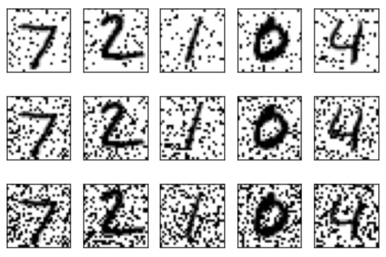


```
def add_noise(raw_data,noise_lv,img_size):
    random.seed(10)
    noise_data=raw_data.copy()
    for i in range(len(noise_data)):
        ran_seq = random.sample([n for n in range(img_size*img_size)], np.int(img_size*img_size*noise_lv)) #隨機選取比例的pixel
        x = noise_data[i].copy().reshape(-1, img_size*img_size)
        x[0, ran_seq]=1 # 取代為255/255
        noise_data[i,:]=x.reshape(-1, img_size, img_size, 1)
    return noise_data


test_x_n10 =  add_noise(test_x,0.1,28)
test_x_n20 =  add_noise(test_x,0.2,28)
test_x_n30 =  add_noise(test_x,0.3,28)
```

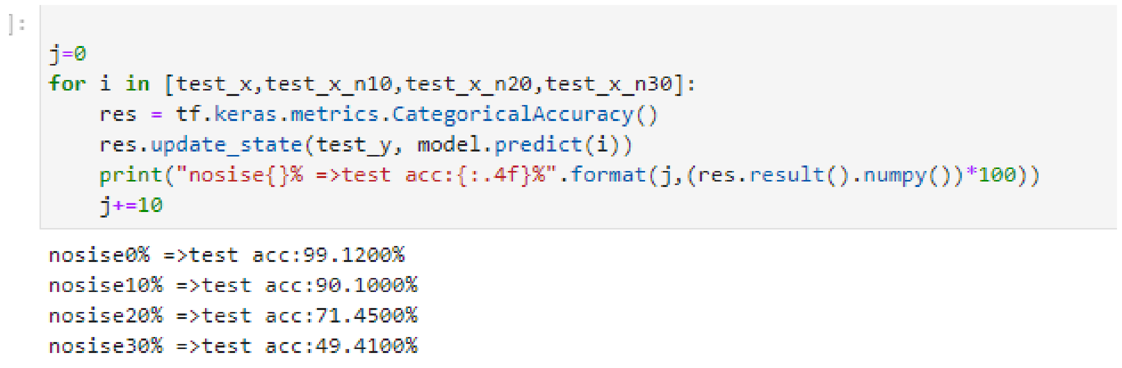
- 視覺化呈現



```
]: import numpy as np
   import matplotlib.pyplot as plt
   amount= 5
   lines = 1
   columns = 5

   fig = plt.figure()

   for i in range(amount):
       ax = fig.add_subplot(lines, columns, 1 + i)
       plt.imshow(test_x_n10[i,:,:], cmap='binary')
       plt.sca(ax)
       ax.set_xticks([])
       ax.set_yticks([])
   fig = plt.figure()
   for i in range(amount):
       ax = fig.add_subplot(lines, columns, 1 + i)
       plt.imshow(test_x_n20[i,:,:], cmap='binary')
       plt.sca(ax)
       ax.set_xticks([])
       ax.set_yticks([])
   fig = plt.figure()
   for i in range(amount):
       ax = fig.add_subplot(lines, columns, 1 + i)
       plt.imshow(test_x_n30[i,:,:], cmap='binary')
       plt.sca(ax)
       ax.set_xticks([])
       ax.set_yticks([])
   plt.show()
```



- 模型結果:



```
]:
j=0
for i in [test_x,test_x_n10,test_x_n20,test_x_n30]:
    res = tf.keras.metrics.CategoricalAccuracy()
    res.update_state(test_y, model.predict(i))
    print("nosise{}% =>test acc:{:.4f}%".format(j,(res.result().numpy())*100))
    j+=10
```

```
nosise0% =>test acc:99.1200%
nosise10% =>test acc:90.1000%
nosise20% =>test acc:71.4500%
nosise30% =>test acc:49.4100%
```

- 現象:

沒有 noise，準確率為 99.12%。10%noise，準確率下降至 90.12%。20%noise，準確率大幅下降至 71.12%。30%noise，準確率大幅下降至 49.41%。

Noise 越多，越會影響模型分類準確率，使分類效果下降。

一點點 noise(%)對準確率的影響較不顯著(準確率跟乾淨資料比下降 9.02%)，但隨著 noise 比例上升，對分類結果的影響是非常大的，如:20% noise 準確率跟乾淨資料比下降 27.67%，與 10%noise 相比下降了 18.65%。30% noise 準確率跟乾淨資料比下降 49.71%，與 20%noise 相比下降了 22.04%。皆比 10%noise 之準確率糟糕，且每增加 10%的 noise，對準確率下降的幅度越大。

3.3 (20%) Please use VGG-16 as a backbone pre-trained on ImageNet to fine-tune your model for the classification and redo Q. 3.1 and Q. 3.2.