真的假不了—基於深度學習辨識面容真偽

Group 9
110034541 李顓廷
110034550 黃喆志
110034566 黃采琳

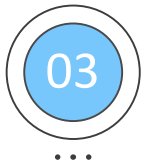
Outline

背景介紹

參數優化

資料蒐集與處理

結論



模型建構

01

背景介紹

研究背景

- 我們從近期的新聞可以得知，隨著換臉技術門檻越來越低，混淆視聽的換臉照片、影片層出不窮，且難以透過肉眼分辨。
- 本小組利用CNN來進行真假臉辨識，期許未來能以此為架構建立一辨識面容真偽的系統。

5W1H

Who	對圖片真偽有疑慮的群眾、偵辦相關案件的人員(警察)
What	辨認圖片人像的真偽
When	當看到圖片時不確定圖中人像的真偽
Where	每個地方
Why	臉部合成技術越來越發達，可防範不法用途
How	深度學習、機器學習、資料分析

02

資料蒐集與處理

資料來源

資料源自於kaggle網站的140k Real and Fake Face資料集

真假臉孔各7萬張圖片數據

資料集已事先將所有數據拆分為訓練、驗證和測試集

資料前處理

Colab導入資料集
將資料集上傳雲端
於colab進行壓縮

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
from zipfile import ZipFile
file_unzip = 'drive/MyDrive/real-vs-fake.zip'
with ZipFile(file_unzip, 'r') as zip:
    zip.extractall()
    print("done")
```

```
done
```

資料前處理

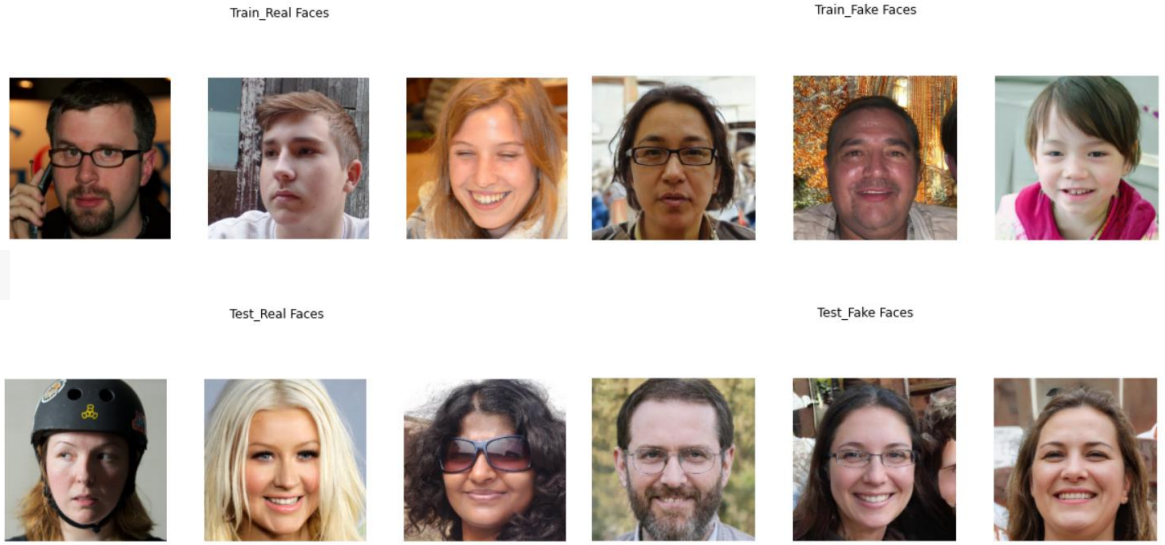
載入相關套件

```
import numpy as np
import pandas as pd
import os
import tensorflow as tf

from keras.preprocessing.image import ImageDataGenerator, load_img
from keras.layers import Conv2D, Dense, BatchNormalization, MaxPooling2D, GlobalAveragePooling2D, Dropout, Flatten

from tensorflow.keras.optimizers import RMSprop,SGD,Adam,Adagrad
from keras import regularizers
from tensorflow.keras.applications import MobileNetV2
import matplotlib.pyplot as plt
from tensorflow.keras.utils import plot_model
```

資料前處理

建立目錄及視覺化訓練及測試集

```
path = '../content/real-vs-fake'
train = '../content/real-vs-fake/train'
test = '/content/real-vs-fake/test'
val = '../content/real-vs-fake/valid'
train_fake = '/content/real-vs-fake/train/fake'
train_real = '/content/real-vs-fake/train/real'
test_fake = '/content/real-vs-fake/test/fake'
test_real = '/content/real-vs-fake/test/real'

def plot_image(path, title):
        plt.figure(figsize=(10,10))
        for i in range(9):
                img = load_img(path +'/' +os.listdir(path)[i])
                plt.subplot(3,3,i+1)
                plt.imshow(img)
                if title=='Fake Faces':
                        plt.title(os.listdir(path)[i][:4])
                plt.suptitle(title)
                plt.axis('off')
        return plt

plot_image(train_real, 'Train_Real Faces').show()
plot_image(train_fake, 'Train_Fake Faces').show()

plot_image(test_real, 'Test_Real Faces').show()
plot_image(test_fake, 'Test_Fake Faces').show()
```



資料前處理

將圖像歸一化

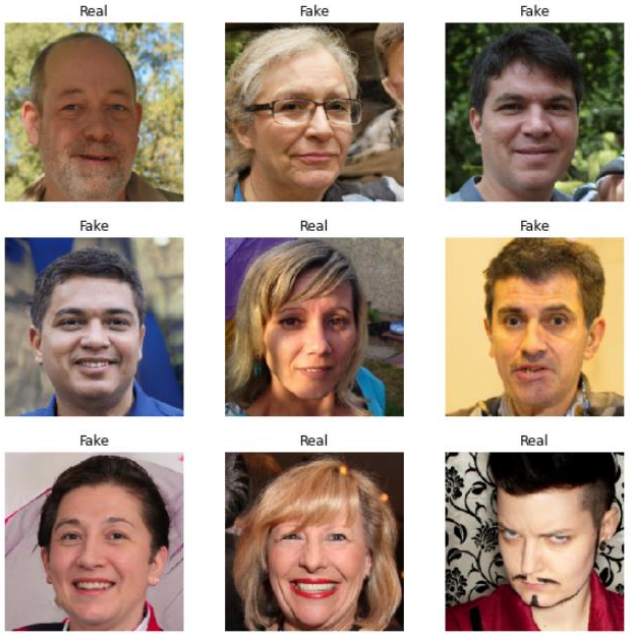
```
datagen  =  ImageDataGenerator(rescale=1./255)

train_set  =  datagen.flow_from_directory(train,
                                          class_mode='binary',
                                          shuffle=True,
                                          target_size=(224,224),
                                          batch_size=64
                                         )
validation_set  =  datagen.flow_from_directory(val,
                                               class_mode='binary',
                                               shuffle=True,
                                               target_size=(224,224),
                                               batch_size=64
                                              )
test_set  =  datagen.flow_from_directory(test,
                                         class_mode='binary',
                                         shuffle=True,
                                         target_size=(224,224),
                                         batch_size=64
                                        )

train_set.class_indices
```

```
Found 100000 images belonging to 2 classes.
Found 20000 images belonging to 2 classes.
Found 20000 images belonging to 2 classes.
{'fake': 0, 'real': 1}
```

資料前處理

視覺化呈現真假圖片標籤

```
plt.figure(figsize=(10, 10))
for i in range(9):
        img, label = train_set.next()
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(img[0])
        if(label[0] == 0.0):
                plt.title("Fake")
        else:
                plt.title("Real")
        plt.axis("off")
```



03

模型建構

模型介紹-CNN

1. **卷積層 Convolution Layer**

   將原始圖片與特定的 Feature Detector(filter)做卷積運算

2. **池化層 Pooling Layer**

   降低了各個特徵圖的維度，但可以保持大分重要的訊息。
   池化層夾在連續的卷積層中間，壓縮數據和參數的量

3. **全連接層 Fully Connected Layer**

   將先前的結果拉直，配合輸出層進行分類

模型建立與訓練-CNN

建立一卷積神經網路模型：

- 第一層layer的filters = 32
- 第二層layer的filters = 64
- 第三層layer的filters = 128
- 進行flatten拉直成1-D陣列
- 輸出fake或real兩類
- Activation function 皆設為relu

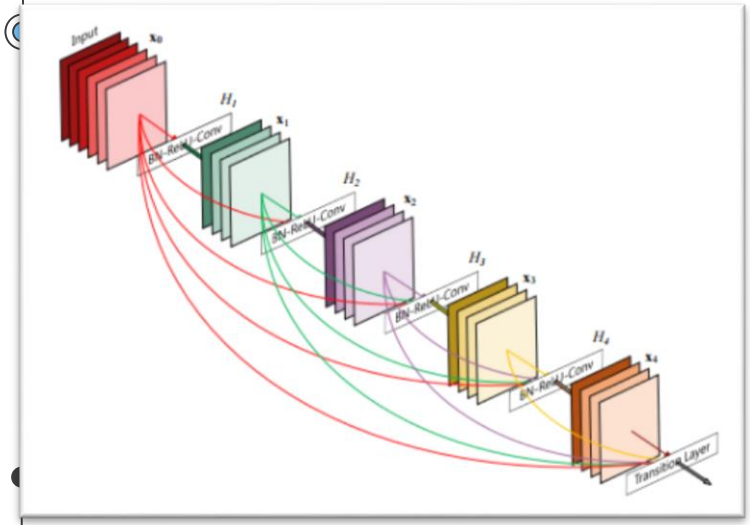
```
model = tf.keras.models.Sequential(
        [
          tf.keras.layers.Conv2D(32, (3,3), padding='same', activation='relu', input_shape=(224, 224, 3)),
          tf.keras.layers.MaxPooling2D(2,2),

          tf.keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
          tf.keras.layers.MaxPooling2D(2,2),

          tf.keras.layers.Conv2D(128, (3,3), padding='same', activation='relu'),
          tf.keras.layers.MaxPooling2D(2,2),

          tf.keras.layers.Flatten(),
          tf.keras.layers.Dense(2)
        ])
model.summary()
model.compile(optimizer= Adam(lr=0.001), loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics = ['accuracy'])
```

模型介紹-Densenet

- 為CNN所延伸發展出的稠密卷積網路
- 此網路不是用很深或很寬的網路來獲得呈現圖像辨識的能力，它是以前饋方式，將每層連結到每個其它層
- 每個層與其後一個層之間，有 L(L + 1)/2 個直接連接 ，並透過特徵的重複使用來得到網絡的隱含訊息，獲得更容易訓練、參數效率更高的稠密模型。

模型介紹-Densenet

Dense block的結構圖



Densenet 架構

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

模型介紹-Densenet

總結

Densenet的優點包含緩解梯度消失問題、加強特徵傳播、大幅減少參數數量；缺點為訓練的過程很佔記憶體，在本研究利用此方法訓練的過程中也有明顯的感受到訓練時間的冗長。

模型建立與訓練-Densenet

建立一稠密卷積神經網路(Densenet)模型:

- 採用 Densenet121架構
- 利用relu及sigmoid作為激活函數

```
#Creating  Model
from  tensorflow.keras.applications  import  DenseNet121
def  get_model():
        densenet  =  DenseNet121(weights='imagenet',
                                  include_top=False,
                                input_shape=(224,224,3)
                                )
        model  =  tf.keras.models.Sequential([densenet,
                                              GlobalAveragePooling2D(),
                                              Dense(512,  activation='relu'),
                                              BatchNormalization(),
                                              Dropout(0.3),
                                              Dense(1,  activation='sigmoid')
                                              ])
        model.compile(optimizer=Adam(lr=0.001),
                                loss='binary_crossentropy',
                                metrics=['accuracy']
                                )

        return  model

spoofnet  =  get_model()
spoofnet.summary()
```

04

參數優化

執行步驟

尋找CNN模型
最適切的epoch

→

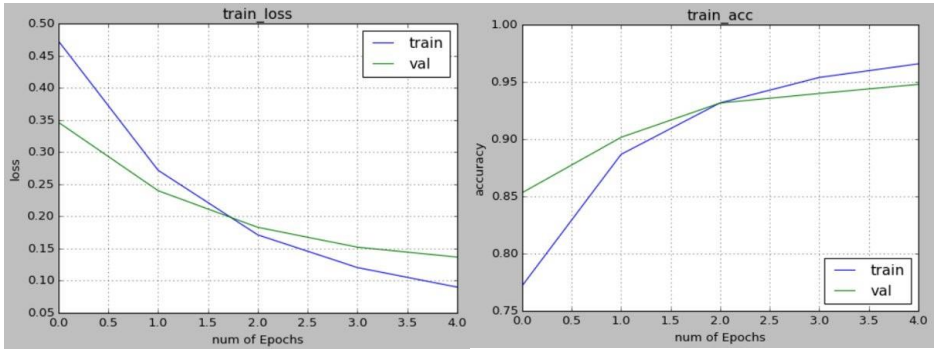
透過實驗設計
超參數優化

→

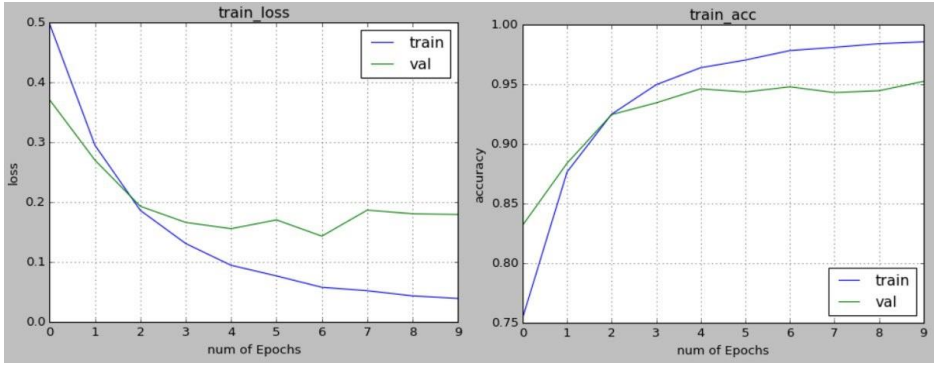
得到優化結果

尋找最適切的epoch

Epoch = 5

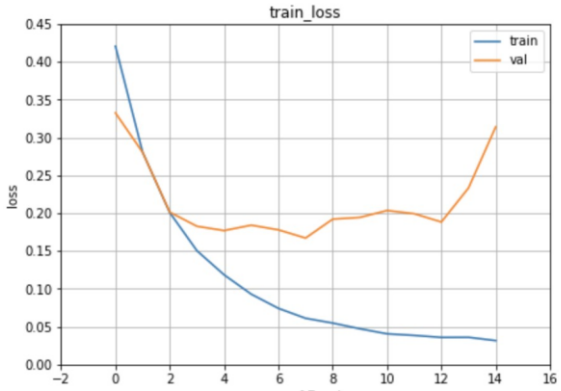


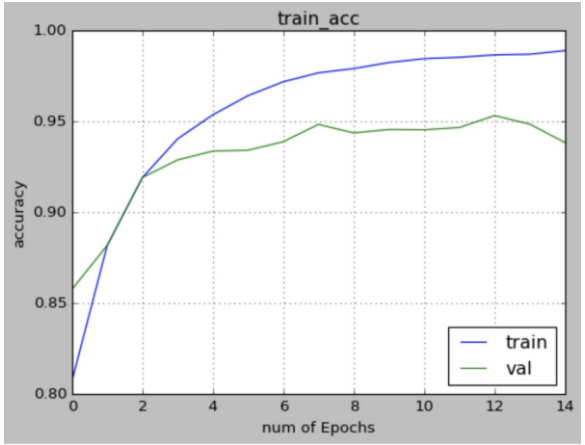
Epoch = 10



尋找最適切的epoch

Epoch=15





由上圖結果可以發現，在epoch=12之後，validation的loss越來越高，具有明顯的overfitting，因此本研究後續進行模型提升、超參數優化，都將epoch設為12來訓練。

實驗設計方法

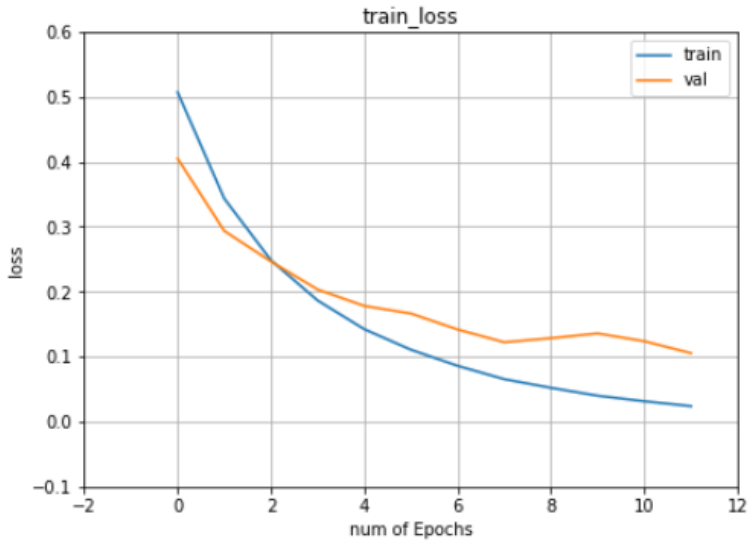
- ✓ 實驗設計是統計學的一種方法
- ✓ 目的 : 減少試驗次數
- ✓ 設定 : 三因子、三水準

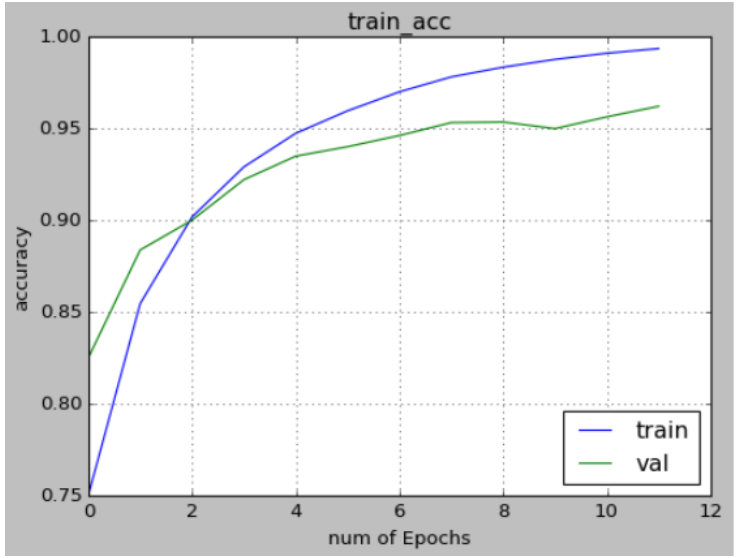
Factor \ Level	Level 1	Level 2	Level 3
Learning rate	0.005	0.001	0.0001
Optimizer	Adam	SGD	Adagrad
Activate function	Relu	Tanh	sigmoid

參數優化結果

	Learning rate	Optimizer	Activate function	train acc	test acc
0	0.001	Adam	Relu	0.9383	0.939
1	0.005	Adam	Relu	0.5004	0.5
2	0.005	SGD	sigmoid	0.5722	0.5928
3	0.005	Adagrad	Tanh	0.9663	0.9319
4	0.001	Adam	sigmoid	0.8878	0.9417
5	0.001	SGD	Tanh	0.7998	0.7962
6	0.001	Adagrad	Relu	0.8175	0.8141
7	0.0001	Adam	Tanh	0.9934	0.9605
8	0.0001	SGD	Relu	0.6325	0.6434
9	0.0001	Adagrad	sigmoid	0.4952	0.5014

實驗7-結果圖

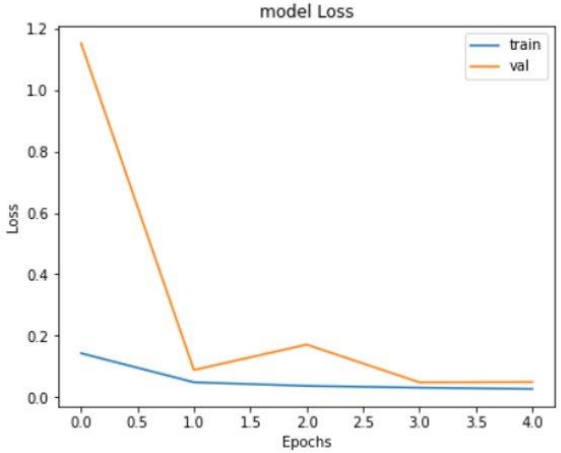

train_loss
train
val
loss
num of Epochs
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−2
0
2
4
6
8
10
12

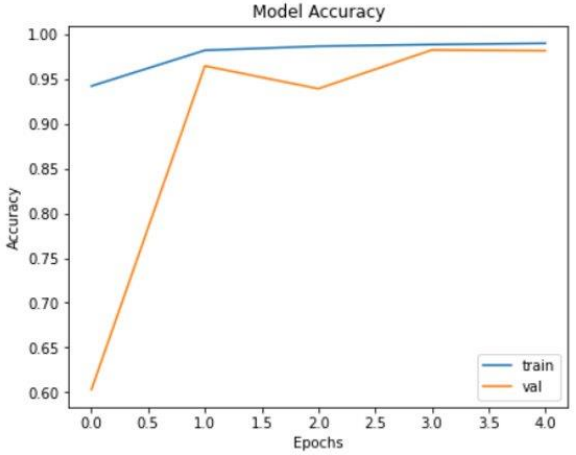

train_acc
train
val
accuracy
num of Epochs
1.00
0.95
0.90
0.85
0.80
0.75
0
2
4
6
8
10
12

Densenet 結果圖

```
history = spoofnet.fit(train_set, validation_data = validation_set, epochs = 5, validation_steps = 100)
```

```
Epoch 1/5
1563/1563 [==============================] - 2090s 1s/step - loss: 0.1429 - accuracy: 0.9422 - val_loss: 1.1515 - val_accuracy: 0.6030
Epoch 2/5
1563/1563 [==============================] - 2043s 1s/step - loss: 0.0487 - accuracy: 0.9821 - val_loss: 0.0885 - val_accuracy: 0.9647
Epoch 3/5
1563/1563 [==============================] - 2042s 1s/step - loss: 0.0368 - accuracy: 0.9867 - val_loss: 0.1711 - val_accuracy: 0.9392
Epoch 4/5
1563/1563 [==============================] - 2052s 1s/step - loss: 0.0310 - accuracy: 0.9886 - val_loss: 0.0484 - val_accuracy: 0.9825
Epoch 5/5
1563/1563 [==============================] - 2053s 1s/step - loss: 0.0268 - accuracy: 0.9900 - val_loss: 0.0494 - val_accuracy: 0.9817
```






```
#Accuracy On test set
_, accu = spoofnet.evaluate(test_set)
print('Final Test Acccuracy = {:.3f}'.format(accu*100))
```

```
313/313 [==============================] - 113s 359ms/step - loss: 0.0477 - accuracy: 0.9830
Final Test Acccuracy = 98.300
```

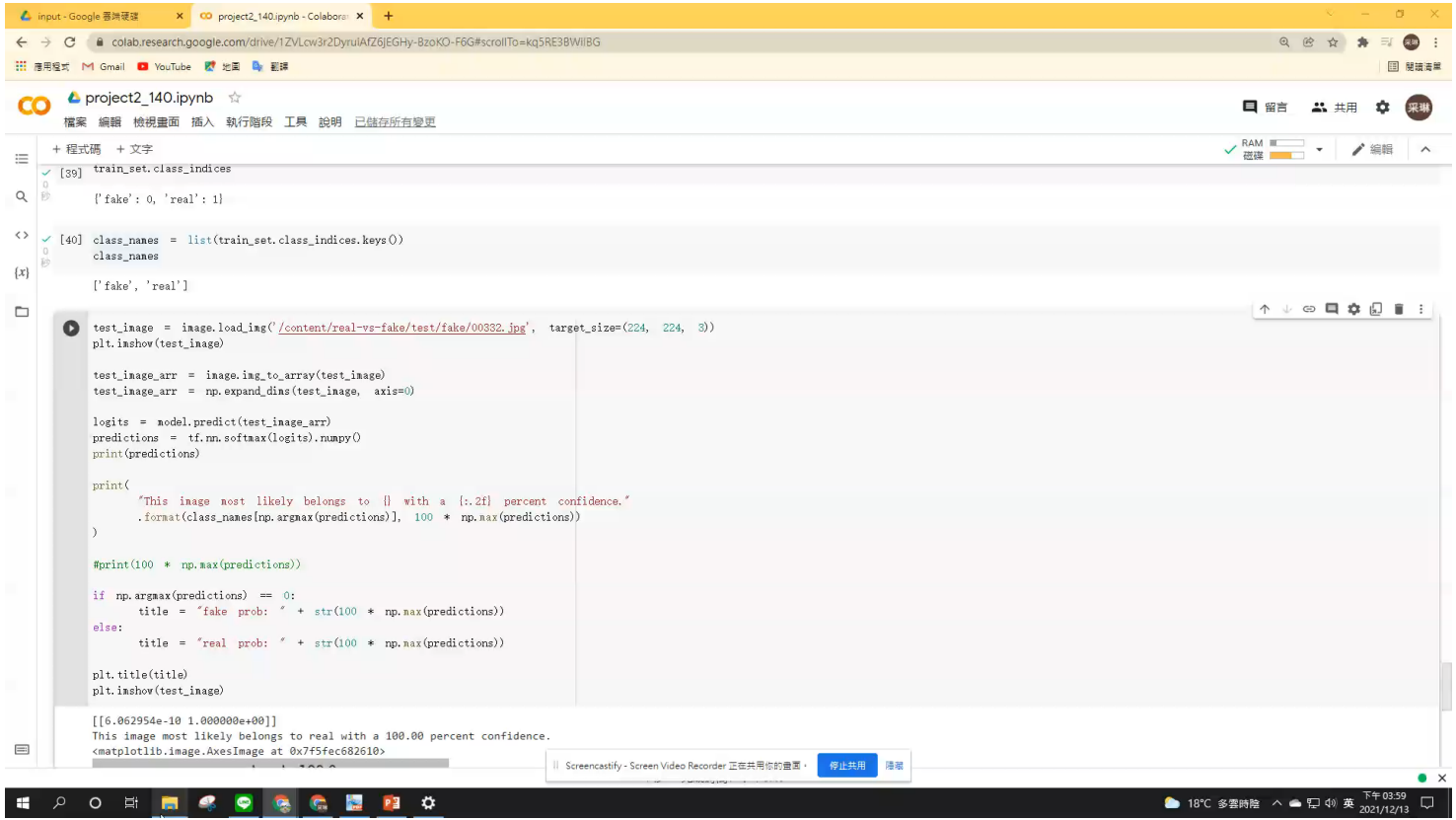
05

結論

研究結果

	CNN	Densenet
全名	卷積神經網路	稠密卷積神經網路
特點	透過卷積和池化層，增進細節辨識能力	加強特徵傳播，特徵重用、減緩梯度消失問題
連接數(L 層)	L 個連接	L（L+1）/ 2 個連接
訓練時間	較短	較長
訓練結果	準確率較低(0.9620)	準確率較高(0.983)

研究結果 – test demo



研究限制與未來研究方向

研究限制

面容真偽辨識僅限於照片，
顯然在現實世界僅辨識照片真偽有泛用性不足的問題，
因為影片比起圖片往往更有混淆大眾視聽的影響力。

未來研究方向

與網站、chatbot做結合，大眾若對收到的圖像有疑慮即可將圖像上傳到所搭建的網站或chatbot，即時確認圖像的真偽。

… … …

Thanks for Listening