國立雲林科技大學電子工程系

人工智慧深度學習

Lab5

ResNet

第十二組

指導教授: 夏世昌 教授

王斯弘 助理教授

組員:四電子三A B10913014 林廷緯

四電子三B B10913158 陳維翔

四電子三B B10913154 曹宸維

中華民國111年11月29日

一、 題目

二、 基本介紹

1. ResNet 網路介紹

2. ResNet 網路實作方式

3. EdgeAI 平台介紹

4. EdgeAI 實作方式

5. 輕量化 tflite 轉換

三、 程式說明

1. ResNet 模型及訓練結果

2. ResNet 模型推理結果

3. h5 檔轉換至 tflite 檔

四、 訓練結果

1. Loss function、Accuracy rate

2. Learning rate

五、 EdgeAI 平台驗證結果

六、 心得與討論

一、 題目

ResNet 架構模型訓練及 EdgeAI 平台驗證

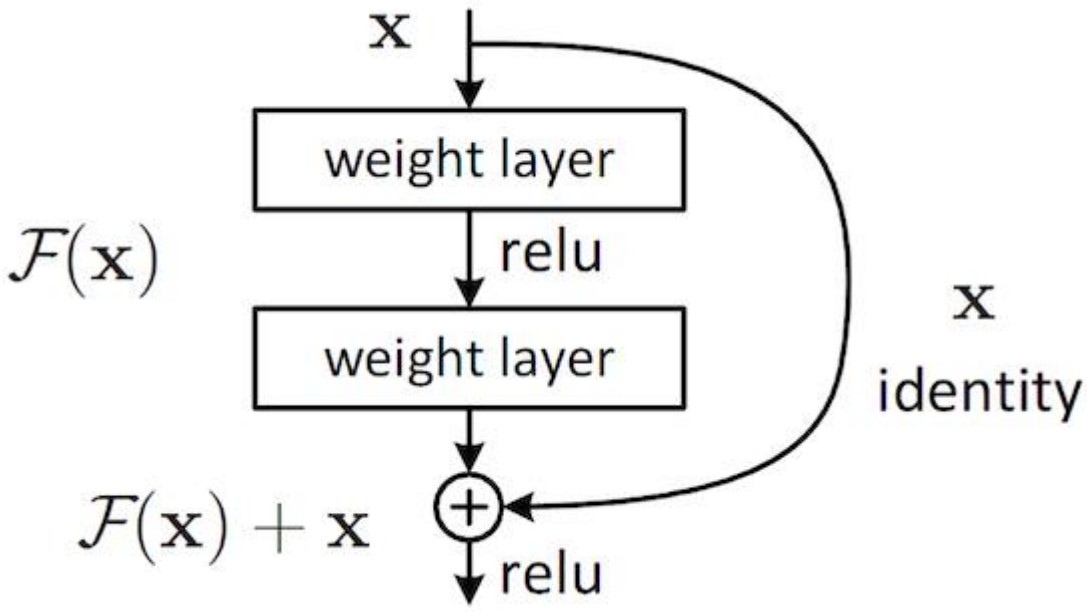
二、 基本介紹

1. ResNet 網路介紹

ResNet 在 2015 年被提出，在 ImageNet 比賽 classification 任務上獲得第一名，因爲它“簡單與實用”並存，之後很多方法都建立在 ResNet50 或者 ResNet101 基礎上完成的，檢測、分割、識別等領域都紛紛使用 ResNet。

2. ResNet 網路實作方式

ResNet 使用了 identity transform。簡單來說就是一個捷徑！這個捷徑會跳過當層，直接成為下兩層 activation function 的 input。



實作時，直接的相加會導致維度上不一致，因此有兩種解決方法。

1. 用 zero padding 增加維度，然後 downsampling(strides=2 的 pooling)，這樣不會增加參數。

2. 就是用 1X1 的 Conv 來 projection，用這個方法就會增加參數。

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

3. EdgeAI 平台介紹

EdgeAI 平台採用樹莓派 3B+，搭配 Google TPU，而將 AI 模型移動至樹莓派時，需使用 TensorFlowLite 檔，壓縮模型大小，以減少樹莓派空間的浪費並加快讀取速度。

4. EdgeAI 實作方式

下載 win32 燒入器，至樹梅派官網下載作業系統，在 win32 燒入器上選擇載下來的作業系統壓縮包，插上磁碟卡並燒入作業系統。

將記憶卡插入樹莓派並開機，經過簡單的初始設定後開啟 Terminal，並依序輸入以下指令，安裝相關套件。

```
1 echo "deb https://packages.cloud.google.com/apt coral-edgetpu-stable main" | sudo tee /etc/apt/sources.list.d/coral-edgetpu.list
2 curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
3 sudo apt-get update
4 sudo apt-get install libedgetpu1-std
5 sudo apt-get install python3-edgetpu
6 sudo apt-get install edgetpu-examples
7 sudo pip3 install jupyterlab opencv-python matplotlib
8 sudo apt-get install python3-pycoral
```

建好環境後，將「tflite 模型」、「label」、「測試圖片」、「py 執行檔」複製到樹莓派後，並下達以下指令，等待一段時間後即可看到影像辨識的結果。

```
python3 py執行檔 --model tflite模型 --labels label --input 測試圖片
```

5. 輕量化 tflite 轉換

利用虛擬機安裝 ubuntu 系統，下達以下指令來建置環境:

```
1 echo "deb https://packages.cloud.google.com/apt coral-edgetpu-stable main" | sudo tee /etc/apt/sources.list.d/coral-edgetpu.list
2 curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
3 sudo apt-get update
4 sudo apt-get install edgetpu-compiler
```

將 h5 檔案轉換成 tflite 檔案並傳輸至虛擬機中，執行以下指令來進行壓縮優化:

```
edgetpu_compiler tflite模型
```

最後再將優化後的模型複製到樹梅派即可。

三、 程式說明

1. ResNet 模型及訓練結果

匯入 tensorflow 並且獲取版本

```
import tensorflow as tf
tf.__version__
```

```
'2.9.2'
```

匯入套件

```
from tensorflow.keras import backend as K
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten,Dense,Dropout,Activation,BatchNormalization,ReLU,add
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Input,GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam,SGD
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot as plt
from tensorflow.keras.utils import plot_model
from tensorflow.keras import optimizers, regularizers
from tensorflow.keras.callbacks import CSVLogger ,EarlyStopping ,ReduceLROnPlateau
```

建立/設定訓練參數

```
DATASET_PATH = 'sample'

IMAGE_SIZE = (224,224)

NUM_CLASSES = 5

BATCH_SIZE = 256

# Epoch 數
NUM_EPOCHS = 60

# 模型輸出儲存的檔案
WEIGHTS_FINAL = 'model-resnet18.h5'
```

建立/設定資料集

```
# 透過 data augmentation 產生訓練與驗證用的影像資料
train_datagen = ImageDataGenerator(rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   channel_shift_range=10,
                                   horizontal_flip=True,
                                   fill_mode='nearest')

train_batches = train_datagen.flow_from_directory(DATASET_PATH + '/train',
                                                  target_size=IMAGE_SIZE,
                                                  interpolation='bicubic',
                                                  class_mode='categorical',
                                                  shuffle=True,
                                                  batch_size=BATCH_SIZE)

valid_datagen = ImageDataGenerator()
valid_batches = valid_datagen.flow_from_directory(DATASET_PATH + '/valid',
                                                  target_size=IMAGE_SIZE,
                                                  interpolation='bicubic',
                                                  class_mode='categorical',
                                                  shuffle=False,
                                                  batch_size=BATCH_SIZE)

# 輸出各類別的索引值
for cls, idx in train_batches.class_indices.items():
    print('Class #{} = {}'.format(idx, cls))
```

定義 Model

```
def block(x,out_filters,k_size=(3,3),downsample=0):
    if(downsample==1):
        x1 = Conv2D(filters=out_filters,kernel_size=(1,1),strides=(2,2),padding='same')(x)
        x=Conv2D(filters=out_filters,kernel_size=k_size,strides=(2,2),padding='same')(x)
    else:
        x1 = x
        x=Conv2D(filters=out_filters,kernel_size=k_size,strides=(1,1),padding='same')(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = Conv2D(filters=out_filters,kernel_size=k_size,strides=(1,1),padding='same')(x)
    x = BatchNormalization()(x)
    x = add([x1,x])
    x = ReLU()(x)
    return x

def resnet18(x):
    x = Conv2D(filters=64,kernel_size=(7,7),strides=(2,2),padding='same')(x)
    x = MaxPooling2D()(x)
    x = block(x,out_filters=64,downsample=0)
    x = block(x,out_filters=64,downsample=0)
    x = block(x,out_filters=128,downsample=1)
    x = block(x,out_filters=128,downsample=0)
    x = block(x,out_filters=256,downsample=1)
    x = block(x,out_filters=256,downsample=0)
    x = block(x,out_filters=512,downsample=1)
    x = block(x,out_filters=512,downsample=0)
    x = GlobalAveragePooling2D()(x)
    x = Dense(5,activation='softmax')(x)
    return x
```

定義 Model 架構及顯示總攬

```
img_input = Input(shape=(IMAGE_SIZE[0],IMAGE_SIZE[1],3))
output = resnet18(img_input)
model = Model(img_input,output)
print(model.summary())
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)] 0
_________________________________________________________________
conv2d (Conv2D)              (None, 112, 112, 64) 9472        input_1[0][0]
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 56, 56, 64)   0         conv2d[0][0]
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 56, 56, 64)   36928       max_pooling2d[0][0]
_________________________________________________________________
batch_normalization (BatchNorma (None, 56, 56, 64)   256         conv2d_1[0][0]
_________________________________________________________________
re_lu (ReLU)                 (None, 56, 56, 64)   0         batch_normalization[0][0]
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 56, 56, 64)   36928       re_lu[0][0]
_________________________________________________________________
batch_normalization_1 (BatchNor (None, 56, 56, 64)   256         conv2d_2[0][0]
_________________________________________________________________
add (Add)                    (None, 56, 56, 64)   0         max_pooling2d[0][0]
```

設定 LOSS 與優化器，訓練模型並儲存結果

```
model.compile(optimizer=Adam(lr=0.001),
                              loss='categorical_crossentropy', metrics=['acc'])

csv_logger = CSVLogger('training.csv')
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2,
                                                  patience=10, min_lr=1e-7)
earlystop = EarlyStopping(monitor ='val_acc', patience=30, verbose=1)
cbks = [csv_logger,reduce_lr]

# 訓練模型
history=model.fit_generator(train_batches,
                              steps_per_epoch = train_batches.samples // BATCH_SIZE,
                              validation_data = valid_batches,
                              validation_steps = valid_batches.samples // BATCH_SIZE,
                              epochs = NUM_EPOCHS,
                              callbacks = cbks
                                                  )
# 儲存訓練好的模型
model.save(WEIGHTS_FINAL)
```

2. ResNet 模型推理結果

匯入套件

```
[ ] from tensorflow.python.keras import backend as K
    from tensorflow.python.keras.models import load_model
    from tensorflow.python.keras.preprocessing import image
    import sys
    import numpy as np
    import glob
```

載入模型檔案

```
[ ]  model  =  load_model('model-resnet18.h5')
```

讀取測試圖片的資料夾

```
files  =  glob.glob('/content/drive/MyDrive/人工智慧深度學習/lab5/*jpeg')
print(files)
```

```
['./test\\bike1.jpeg', './test\\bike2.jpeg', './test\\car1.jpeg', './test\\cat1.jpeg', '
```

AI 模型推理結果

```
cls_list  =  ['bike',  'car',  'cats',  'doge',  'ox']

for  f  in  files:
        img  =  image.load_img(f,  target_size=(224,  224))
        if  img  is  None:
                continue
        x  =  image.img_to_array(img)
        x  =  np.expand_dims(x,  axis  =  0)
        pred  =  model.predict(x)[0]
        top_inds  =  pred.argsort()[::-1][:5]
        print(f)
        for  i  in  top_inds:
                print('    {:.3f}    {}'.format(pred[i],  cls_list[i]))
```

```
./test\bike1.jpeg
  0.999  bike
  0.001  car
  0.000  ox
  0.000  doge
  0.000  cats
./test\bike2.jpeg
  0.609  bike
  0.333  car
  0.050  ox
  0.008  cats
  0.000  doge
```

3. h5 檔轉換至 tflite 檔

匯入相關套件

```
import  tensorflow  as  tf
#from  tensorflow.keras  import  Model
import  numpy  as  np
import  glob
tf.__version__
```

```
'2.9.2'
```

轉換副程式

```
IMAGE_SIZE = 224
def representative_data_gen():
        dataset_list=glob.glob('./sample/*/*/*')
        dataset_list_index=np.random.choice(range(len(dataset_list)),100)
        for i in range(100):
                image = dataset_list[dataset_list_index[i]]
                print(image)
                image = tf.io.read_file(image)
                image = tf.io.decode_jpeg(image,channels=3)
                image = tf.image.resize(image,[IMAGE_SIZE, IMAGE_SIZE])
                image = tf.cast(image,tf.float32)
                image = tf.expand_dims(image,0)
                with tf.Session() as sess:
                        image = sess.run(image)
                yield [image]
```

開始轉換模型

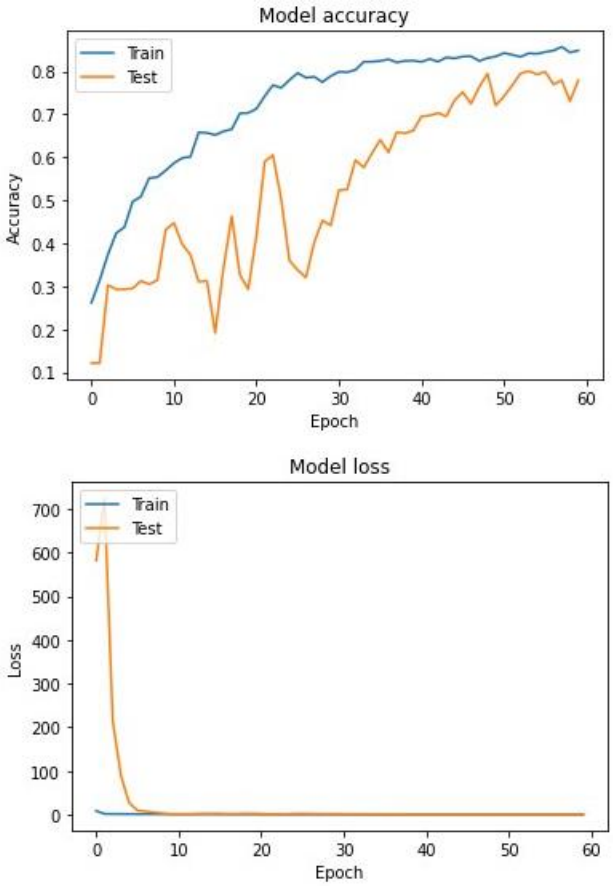
```
if tf.__version__[0]=='2':
        converter = tf.lite.TFLiteConverter.from_keras_model(save_keras_model)
elif tf.__version__[0]=='1':
        converter = tf.lite.TFLiteConverter.from_keras_model_file('./model-resnet18.h5')
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
converter.inference_input_type = tf.uint8
converter.inference_output_type = tf.uint8
converter.representative_dataset = representative_data_gen
tflite_model = converter.convert()
```

寫入模型

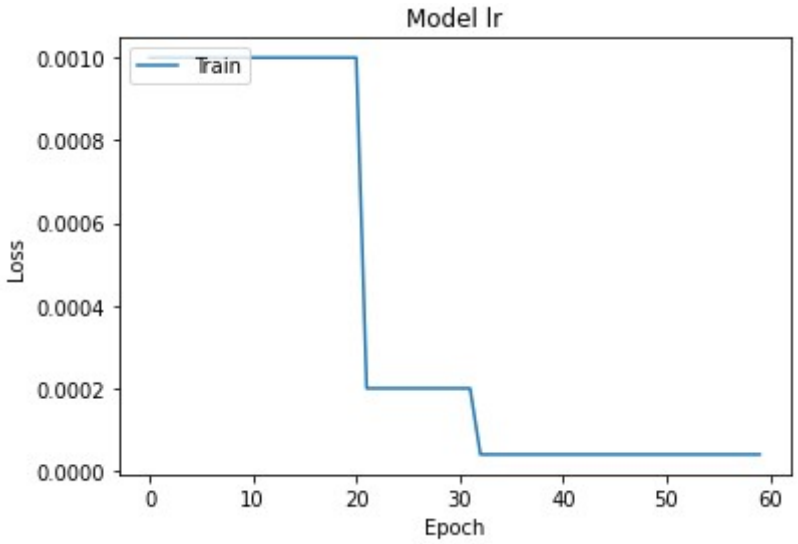
```
[ ] with open('resnet18.tflite','wb') as f:
        f.write(tflite_model)
```

四、 訓練結果

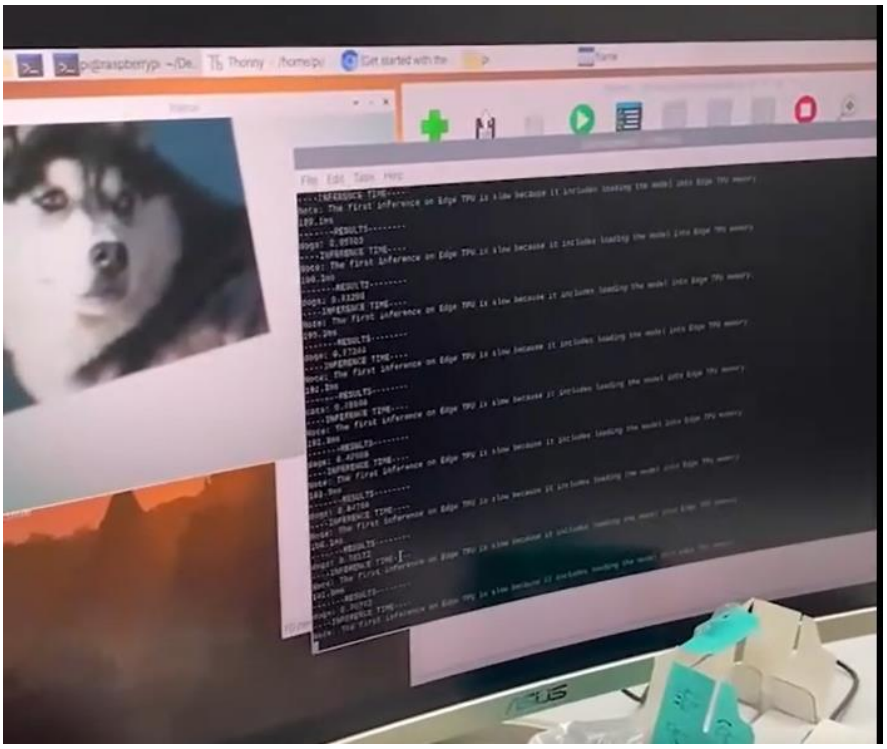
1. Loss function、Accuracy rate

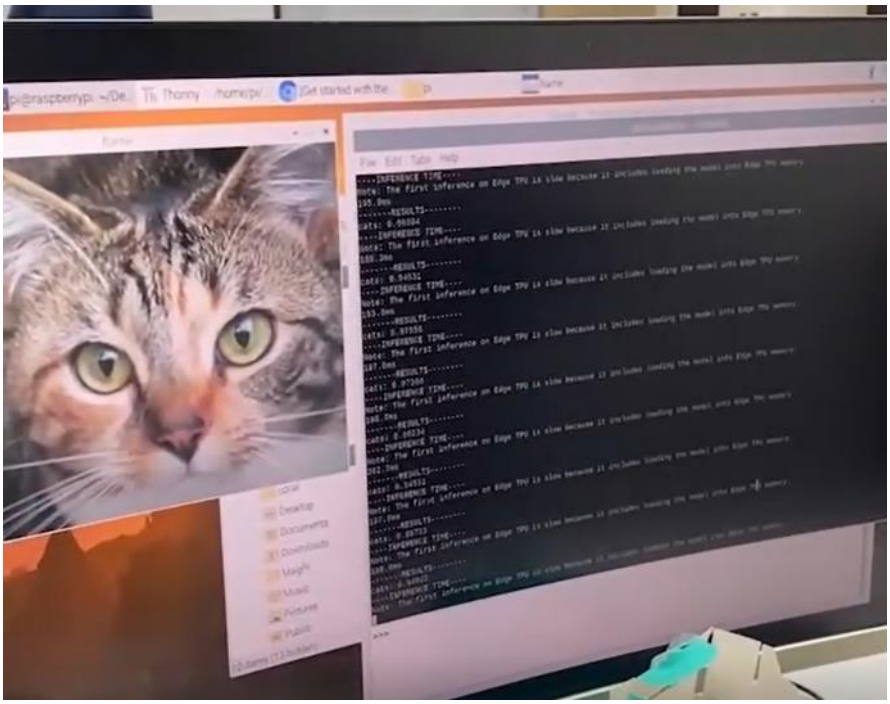


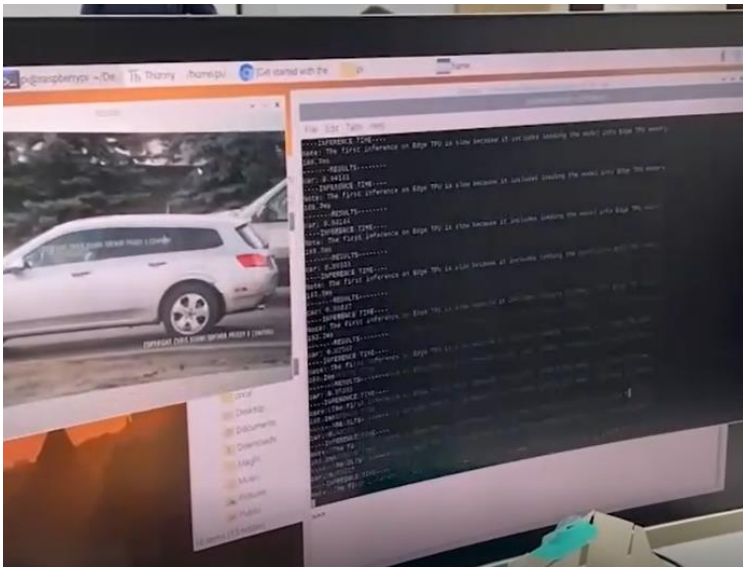
2. Learning rate

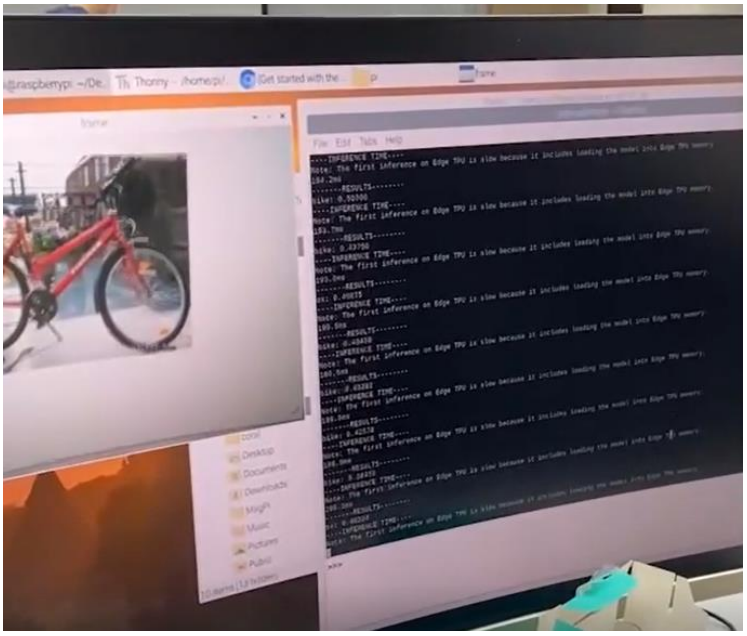


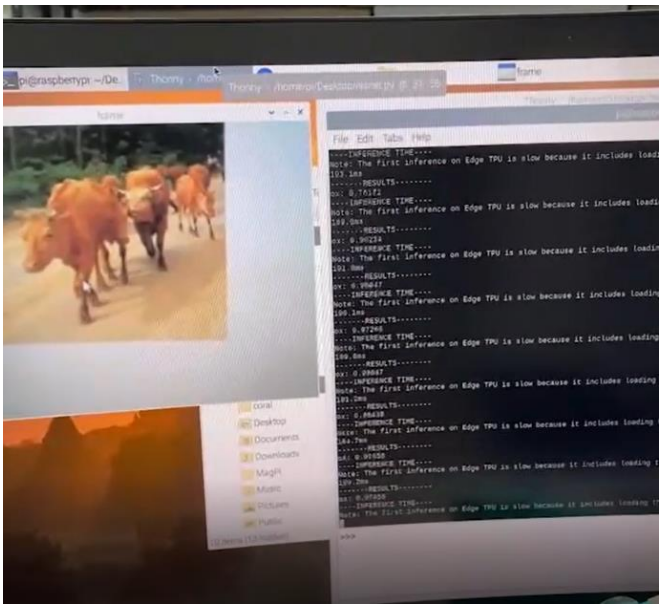
五、 EdgeAI 平台驗證結果











六、 心得與討論

陳維翔:

這次的 lab 讓我了解到 ResNet 要怎麼去使用，他能用在什麼地方，這次的實作中，遇到了些之前的問題，不過還是順利解決了，可能是訓練沒弄好，最後驗證的結果好壞參半，有些可以有些怪怪的，希望下次能更好。

曹宸維:

這次的 lab 用 ResNet 分析了五種類別的圖片並判斷被攝像頭拍到的圖是否屬於其中一種類別，我覺得很有趣，也學到了很多，雖然結果沒到很好，但還是有判斷出來，我想只要把訓練用的圖片增加就可以改善這個問題了。

林廷緯:

這次的 lab 用的是 ResNet，很吃顯卡效能，用自己的電腦跑得超久，只能靠學校的電腦，程式的部分跟之前的很像，所以沒什麼問題，期望下次能做得更快。