

國立雲林科技大學電子工程系
人工智慧深度學習

Lab4
LeNet

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 14 日

一、 題目

二、 基本介紹

1. LeNet 網路介紹
2. LeNet 網路實作方式
3. LeNet 平台介紹
4. EdgeAI 實作方式
5. 輕量化 tflite 轉換

三、 程式說明

1. LeNet 模型及訓練結果
2. LeNet 模型推理結果
3. h5 檔轉換至 tflite 檔

四、 訓練結果

1. Loss function
2. Accuracy rate

五、 EdgeAI 平台驗證結果

六、 心得與討論

一、 題目

LeNet 架構模型訓練及 EdgeAI 平台驗證

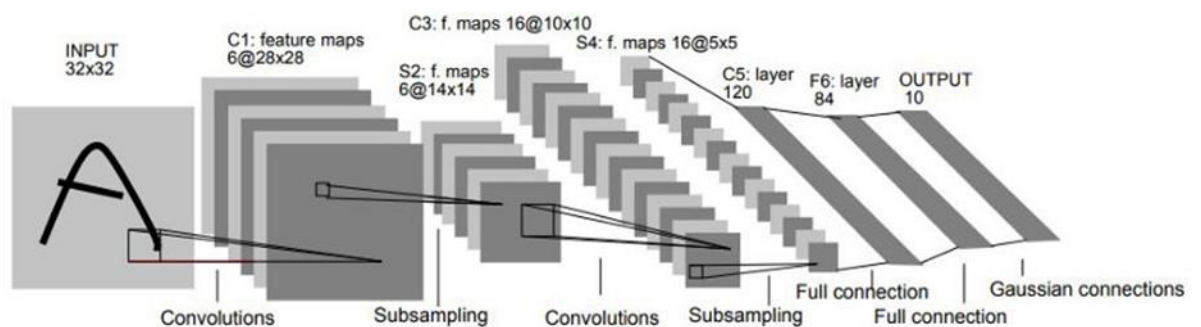
二、 基本介紹

1. LeNet 網路介紹

LeNet 是由 Yann LeCun 團隊提出的網路架構，是卷積神經網路的始祖。早期用來辨識手寫數字圖像，架構簡單且直覺，適合初學入門

2. LeNet 網路實作方式

共有 7 層架構:C1、S2、C3、S4、C5、F6、OUTPUT，字母代表神經層種類(C:卷積層、S:池化層、F:全連接層)，數字代表層數為第幾層。



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

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

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. LeNet 模型及訓練結果

匯入 tensorflow 並且獲取版本

```
import tensorflow as tf
import numpy as np

tf.__version__

'2.9.2'
```

建立/設定訓練參數和資料集

```
num_class = 10
batch_size = 2048
epochs = 100
iterations = 30

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

設定資料處理


```
x_train = x_train / 255
x_test = x_test / 255
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], x_train.shape[2], 1)) # (60000, 28, 28, 1)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], x_test.shape[2], 1)) # (10000, 28, 28, 1)
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)
```

定義 Convolution 和 pooling 層

```
[3] def conv(x, filter, size):
    return tf.keras.layers.Conv2D(filters=filter, kernel_size=size)(x)

[4] def maxpooling(x):
    return tf.keras.layers.MaxPool2D(padding='same', strides=2)(x)
```

定義 Model

```
 def lenet(x):
    x=conv(x, 6, (5, 5))
    x=maxpooling(x)
    x=conv(x, 16, (5, 5))
    x=maxpooling(x)
    x=tf.keras.layers.Flatten()(x)
    x=tf.keras.layers.Dense(120)(x)
    x=tf.keras.layers.Dense(84)(x)
    x=tf.keras.layers.Dense(10, activation='softmax')(x)
```

定義 Model 架構及顯示總覽

```
img_input=tf.keras.Input(shape=(28, 28, 1))
output=lenet(img_input)
model=tf.keras.Model(img_input, output)
print(model.summary())
```

Model: "model_5"

Layer (type)	Output Shape	Param #
input_8 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d_10 (Conv2D)	(None, 24, 24, 6)	156
max_pooling2d_10 (MaxPooling2D)	(None, 12, 12, 6)	0
conv2d_11 (Conv2D)	(None, 8, 8, 16)	2416
max_pooling2d_11 (MaxPooling2D)	(None, 4, 4, 16)	0
flatten_5 (Flatten)	(None, 256)	0
dense_15 (Dense)	(None, 120)	30840
dense_16 (Dense)	(None, 84)	10164
dense_17 (Dense)	(None, 10)	850
Total params: 44,426		
Trainable params: 44,426		
Non-trainable params: 0		
None		

設定 LOSS 與優化器，並儲存結果

```
sgd=tf.keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=sgd,loss='categorical_crossentropy',metrics=['acc'])
history=model.fit(x=x_train,y=y_train,batch_size=batch_size,epochs=epochs,steps_per_epoch=iterations,validation_data=(x_test,y_test))
```

```
model.save('/content/minst_lenet.h5')
```

2. LeNet 模型推理結果

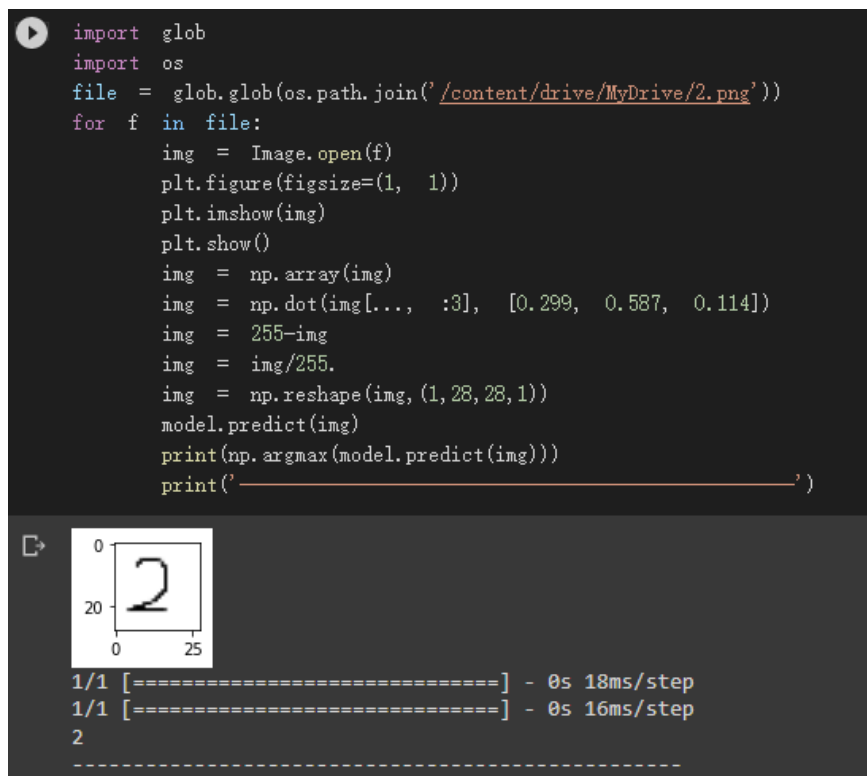
載入模型檔案

```
model = tf.keras.models.load_model('/content/minst_lenet.h5')  
  
[36] from google.colab import drive  
      drive.mount('/content/drive')
```

讀取測試圖片的資料夾



AI 模型推理結果



3. h5 檔轉換至 tflite 檔

匯入相關套件

```
import tensorflow as tf
import numpy as np
import glob
tf.__version__
```

'2.9.2'

轉換副程式

```
IMAGE_SIZE = 28

def color_preprocessing(x_train, x_test):
    x_train = x_train.astype('float32')
    x_test = x_test.astype('float32')
    x_train = x_train/255.
    x_test = x_test/255.
    return x_train, x_test

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)
x_train, x_test = color_preprocessing(x_train, y_train)

x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], x_train.shape[2], 1))
```

開始轉換模型

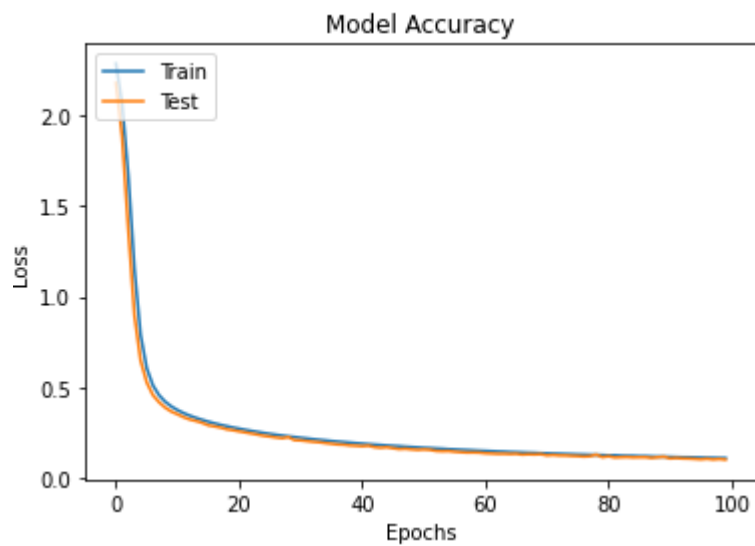
```
keras_model = tf.keras.models.load_model("/content/minst_lenet.h5")
converter = tf.lite.TFLiteConverter.from_keras_model(keras_model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
converter.inference_input_type = tf.uint8
converter.inference_output_type = tf.uint8
converter.representative_dataset = representative_data_gen
tflite_model = converter.convert()
```

寫入模型

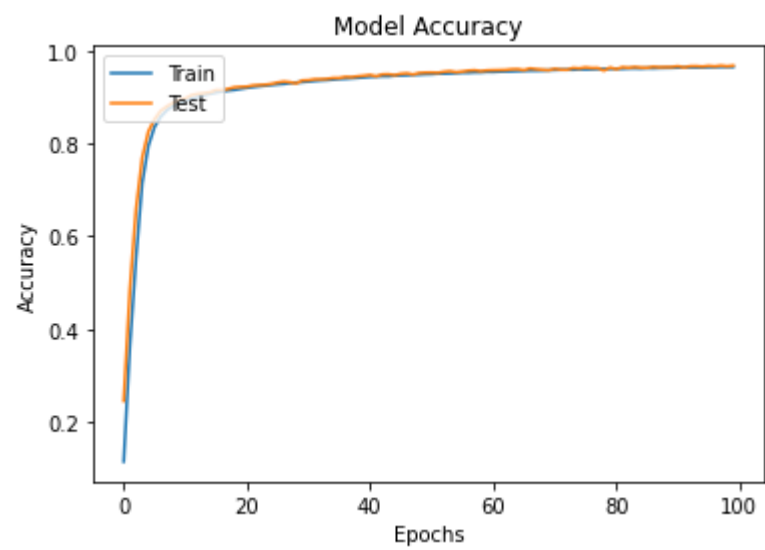
```
with open('mnist115.tflite', 'wb') as f:
    f.write(tflite_model)
```

四、 訓練結果

1. Loss function

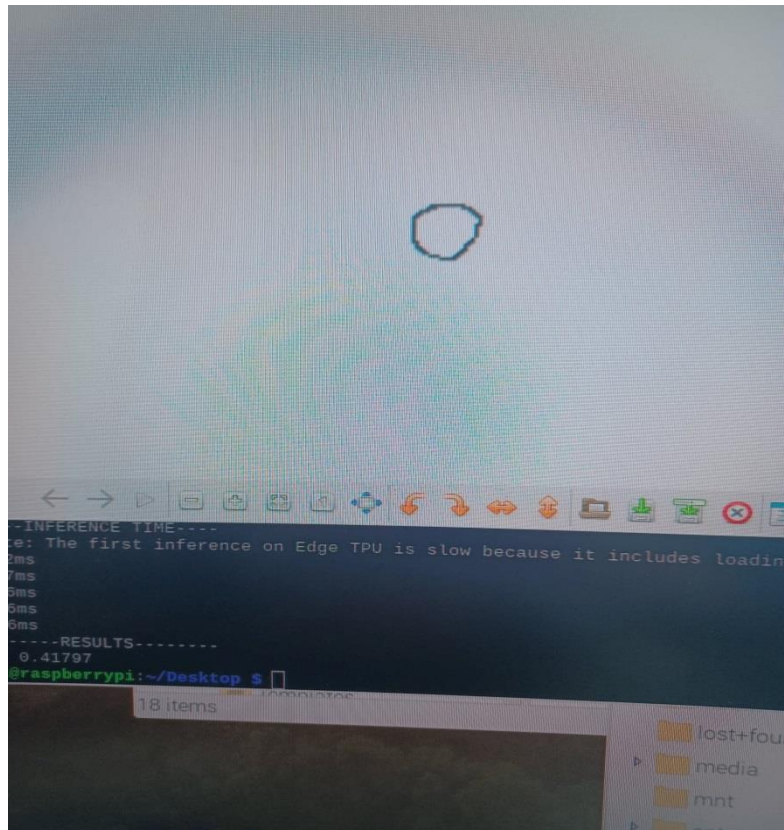


2. Accuracy rate

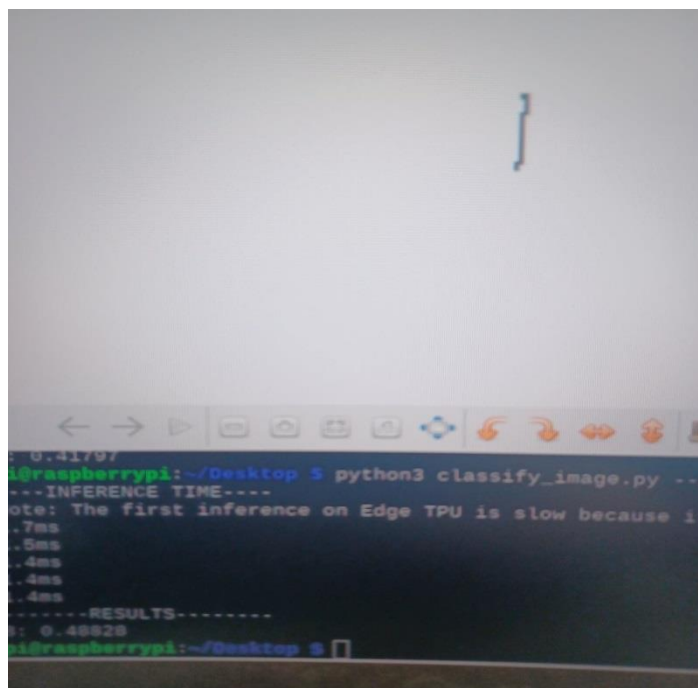


五、 EdgeAI 平台驗證結果

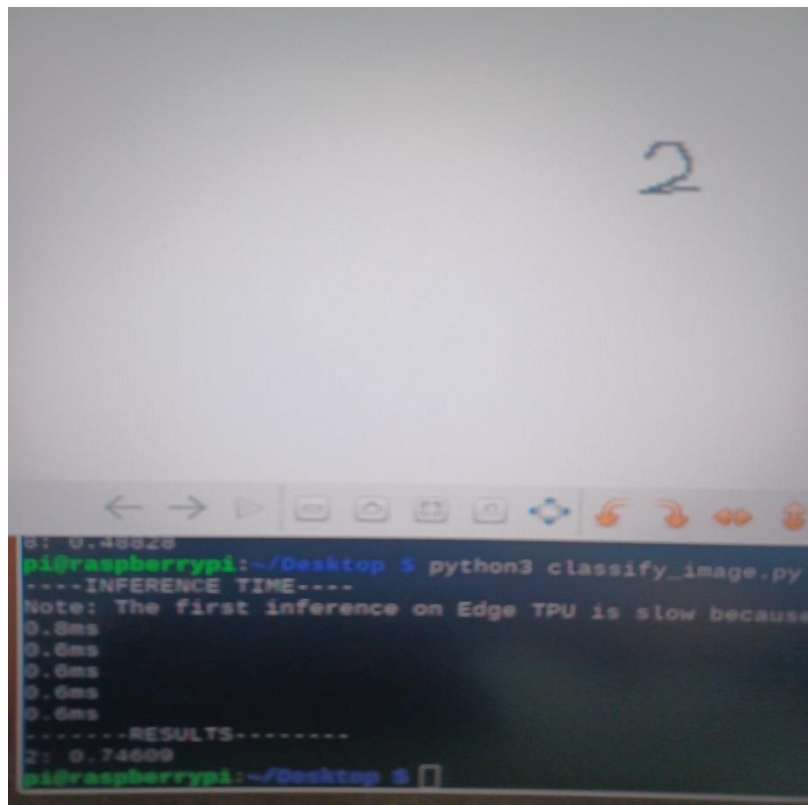
0



1



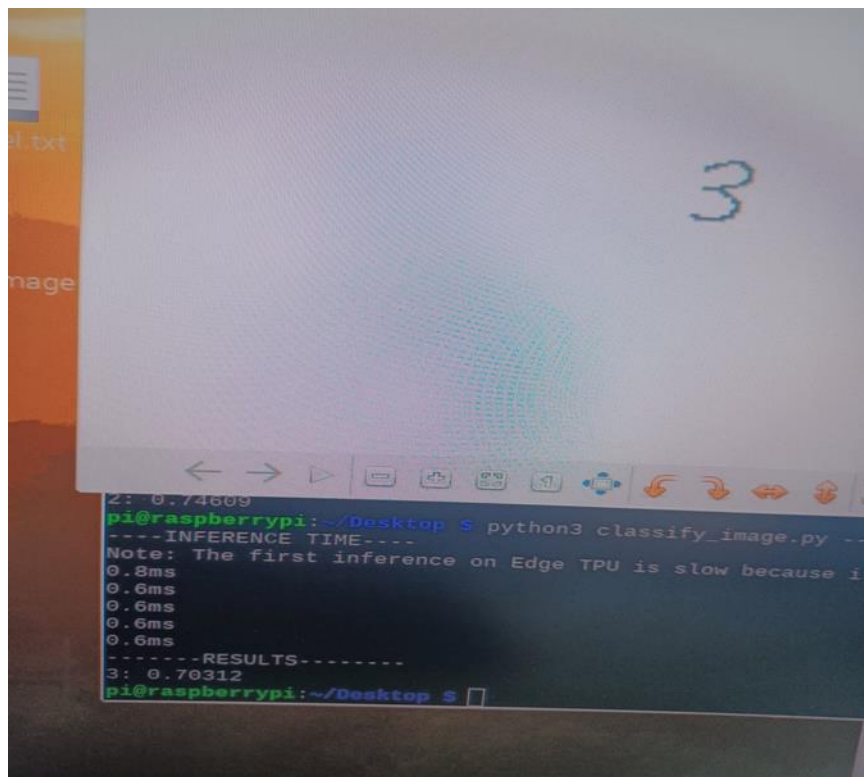
2



A terminal window on a Raspberry Pi. The top part of the window shows a grayscale image of a handwritten digit '2'. Below the image is a terminal window with the following text:

```
0: 0.48828
pi@raspberrypi:~/Desktop $ python3 classify_image.py
----INFERENCE TIME----
Note: The first inference on Edge TPU is slow because
0.8ms
0.6ms
0.6ms
0.6ms
0.6ms
-----RESULTS-----
2: 0.74609
pi@raspberrypi:~/Desktop $
```

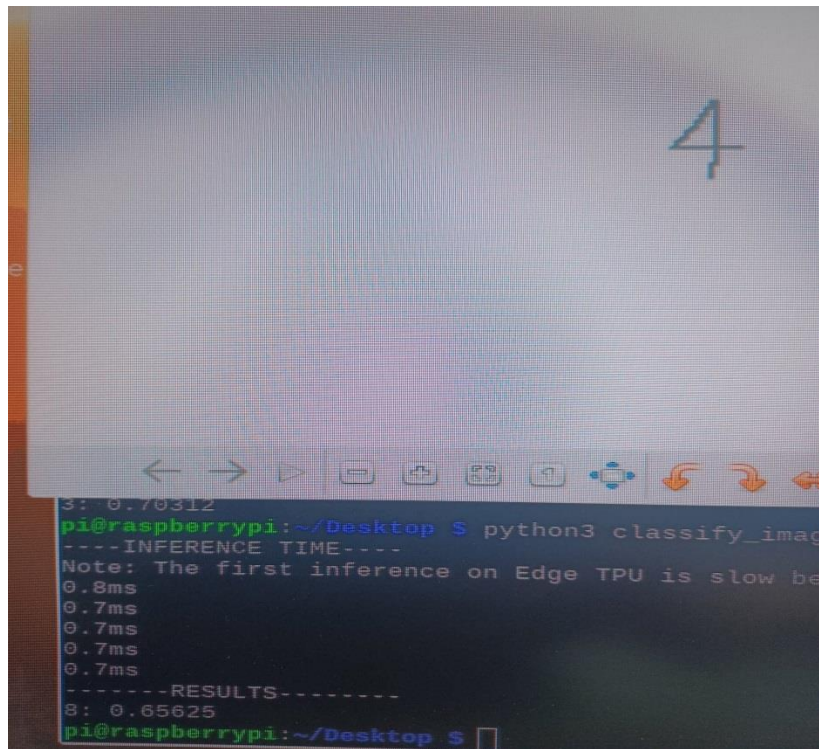
3



A terminal window on a Raspberry Pi. The top part of the window shows a grayscale image of a handwritten digit '3'. Below the image is a terminal window with the following text:

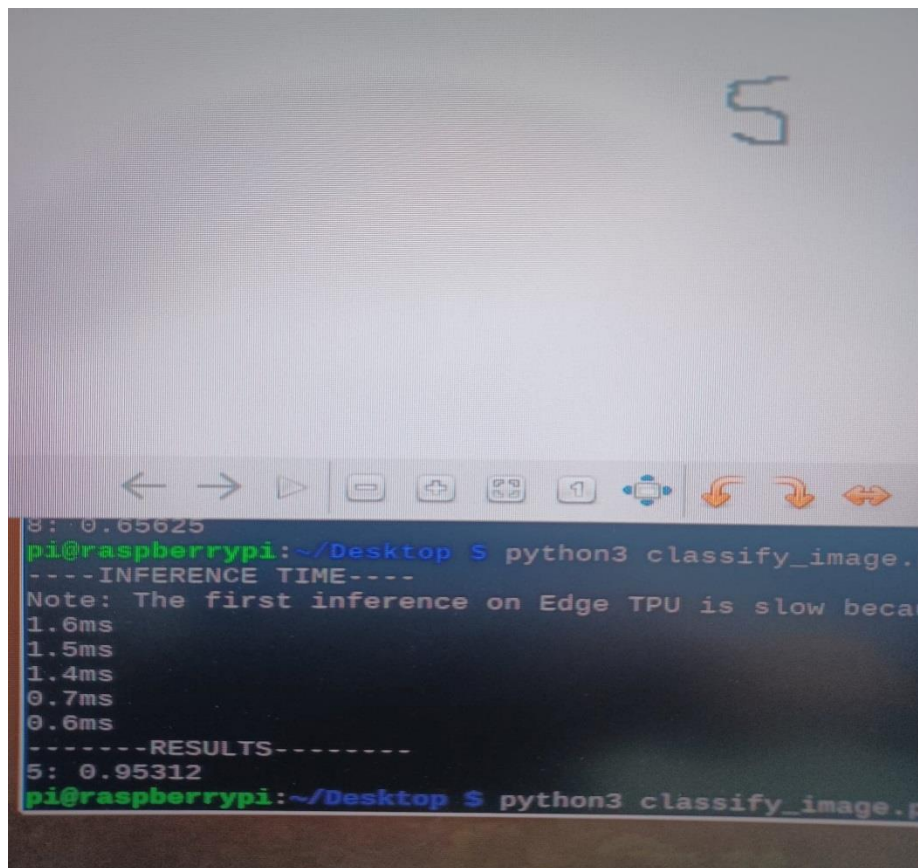
```
2: 0.74609
pi@raspberrypi:~/Desktop $ python3 classify_image.py --
----INFERENCE TIME----
Note: The first inference on Edge TPU is slow because it
0.8ms
0.6ms
0.6ms
0.6ms
0.6ms
-----RESULTS-----
3: 0.70312
pi@raspberrypi:~/Desktop $
```

4



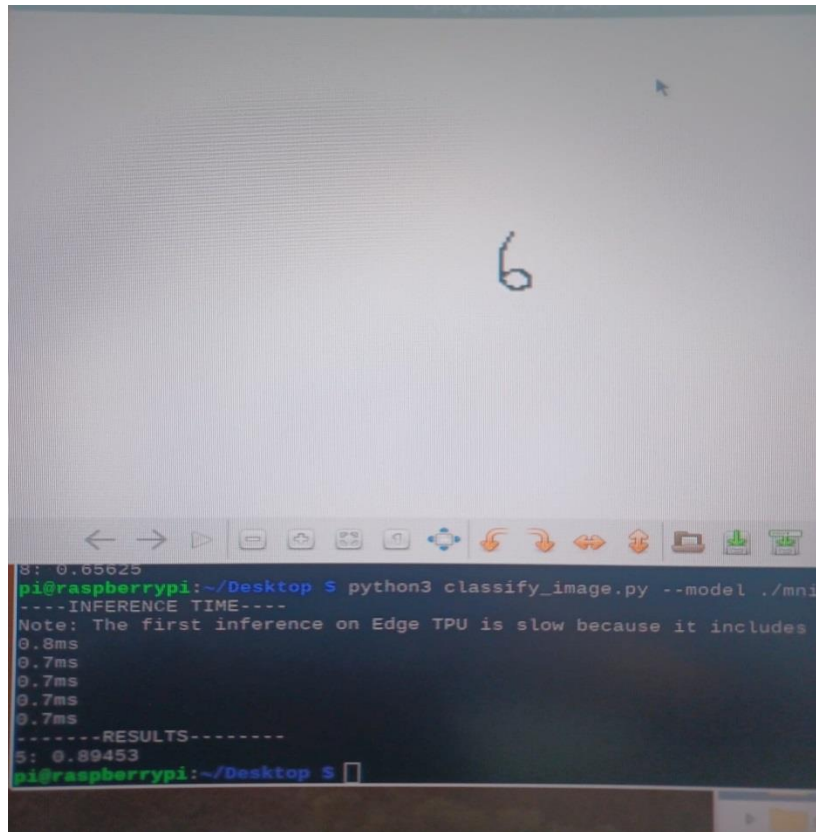
```
3: 0.70312
pi@raspberrypi:~/Desktop $ python3 classify_image.py
----INFERENCE TIME----
Note: The first inference on Edge TPU is slow because the model is not optimized.
0.8ms
0.7ms
0.7ms
0.7ms
0.7ms
-----RESULTS-----
8: 0.65625
pi@raspberrypi:~/Desktop $
```

5



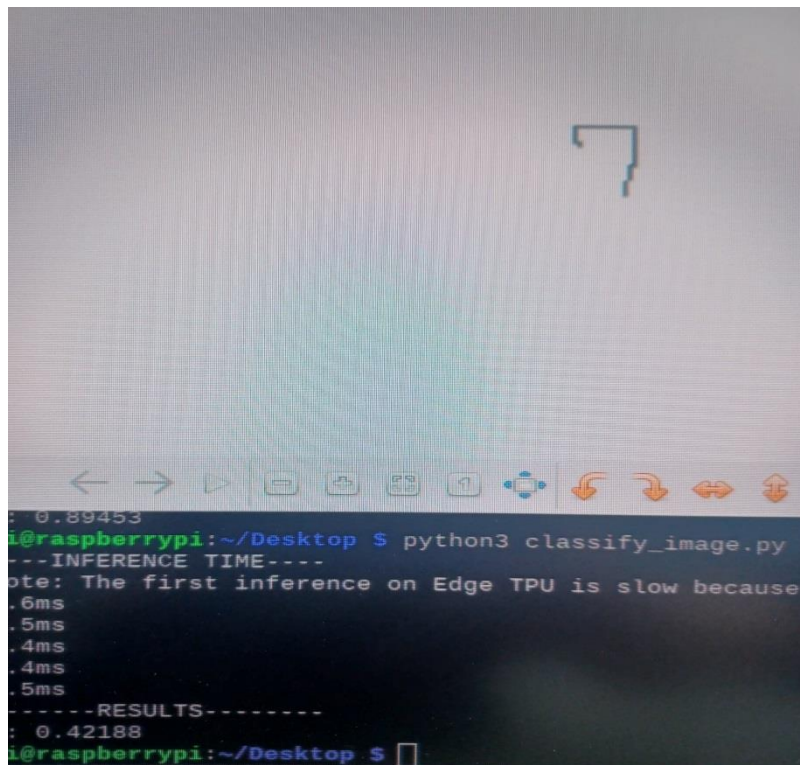
```
8: 0.65625
pi@raspberrypi:~/Desktop $ python3 classify_image.py
----INFERENCE TIME----
Note: The first inference on Edge TPU is slow because the model is not optimized.
1.6ms
1.5ms
1.4ms
0.7ms
0.6ms
-----RESULTS-----
5: 0.95312
pi@raspberrypi:~/Desktop $ python3 classify_image.py
```


6



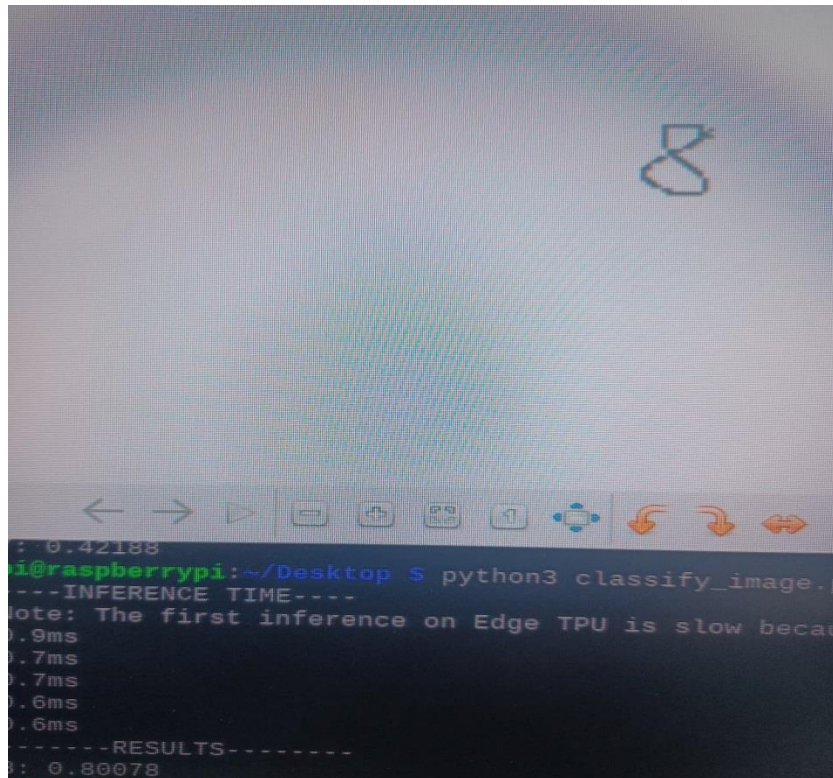
```
8: 0.65625
pi@raspberrypi:~/Desktop $ python3 classify_image.py --model ./mn1
----INFERENCE TIME----
Note: The first inference on Edge TPU is slow because it includes
0.8ms
0.7ms
0.7ms
0.7ms
0.7ms
-----RESULTS-----
S: 0.89453
pi@raspberrypi:~/Desktop $
```

7



```
: 0.89453
i@raspberrypi:~/Desktop $ python3 classify_image.py
---INFERENCE TIME---
ote: The first inference on Edge TPU is slow because
.6ms
.5ms
.4ms
.4ms
.5ms
-----RESULTS-----
: 0.42188
i@raspberrypi:~/Desktop $
```

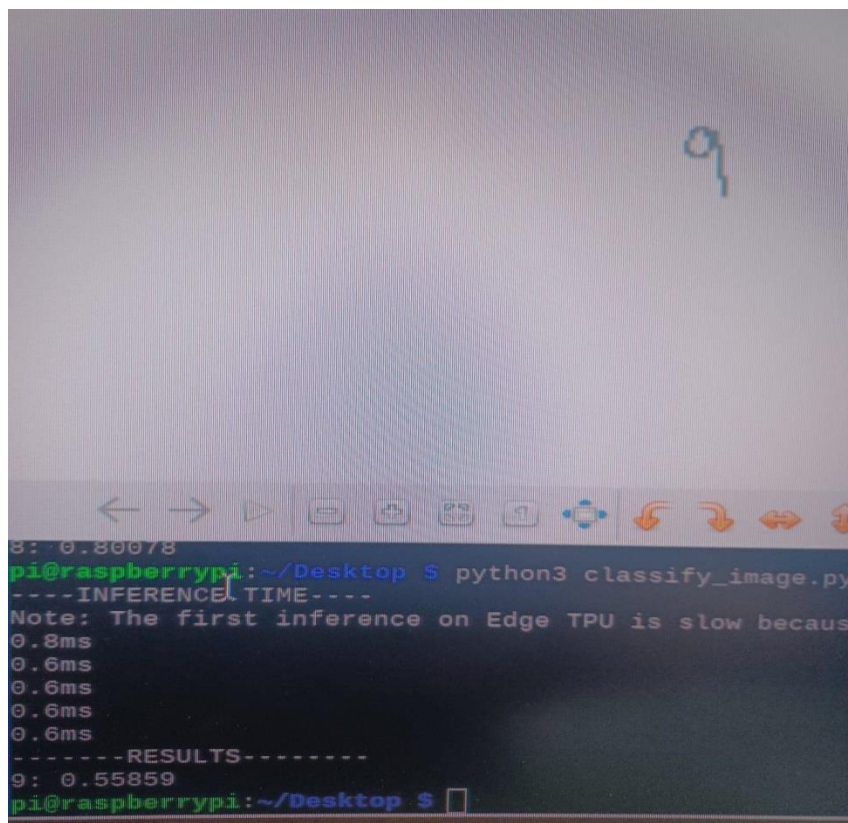
8



A screenshot of a terminal window on a Raspberry Pi. The terminal shows the execution of a script named `classify_image.py`. The output includes inference times and classification results for the digit '8'. The terminal window has a standard Linux prompt and a toolbar at the bottom.

```
pi@raspberrypi:~/Desktop $ python3 classify_image.py
---INFERENCE TIME---
Note: The first inference on Edge TPU is slow because of TensorFlow's calibration step.
0.9ms
0.7ms
0.7ms
0.6ms
0.6ms
-----RESULTS-----
8: 0.80078
```

9



A screenshot of a terminal window on a Raspberry Pi. The terminal shows the execution of a script named `classify_image.py`. The output includes inference times and classification results for the digit '9'. The terminal window has a standard Linux prompt and a toolbar at the bottom.

```
pi@raspberrypi:~/Desktop $ python3 classify_image.py
---INFERENCE TIME---
Note: The first inference on Edge TPU is slow because of TensorFlow's calibration step.
0.8ms
0.6ms
0.6ms
0.6ms
0.6ms
-----RESULTS-----
9: 0.55859
pi@raspberrypi:~/Desktop $
```

六、心得與討論

陳維翔：

這次的 lab 做完滿有成就感的，也讓我感到有趣，原來人工 AI 是這麼判斷影像的，讓我對之後要學的東西感到更有興趣了，一開始看到那麼多程式讓人感到害怕，不過成功做完讓人感到滿滿的成就感

曹宸維：

這次使用樹莓派和訓練好的 AI 進行邊緣運算，不知為甚麼訓練出來的正確率沒那麼穩定，也許是模型沒寫好之類的，不過總的來說還是成功的，這次的 lab 也讓我們更了解 lenet 怎麼去使用

林廷緯：

透過這次的 lab 我學到了很多關於 lenet 以及各種 AI 運算的事，這讓我以後在接觸到其他相關物件時可以更快速的去明白並使用