

Lab-2 LeNet

第四組 - 四資管AI三A - B11123224 - 余雨芹



TABLE OF CONTENT

01. 模型介紹
02. PC端 模型訓練&推理
03. Edge端 操作流程
04. Edge端 驗證結果
05. 遇到問題與解決
06. 心得

01



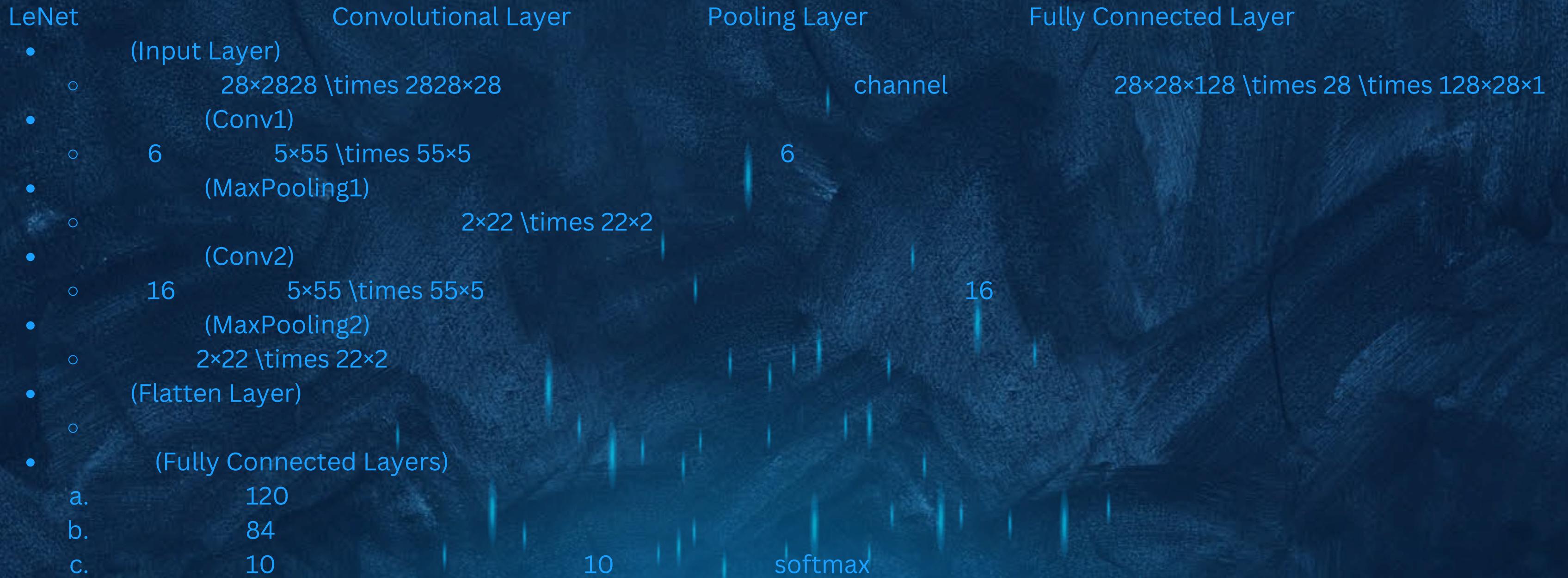
MNIST

CNN

--LeNet

LeNet

Yann LeCun



MNIST

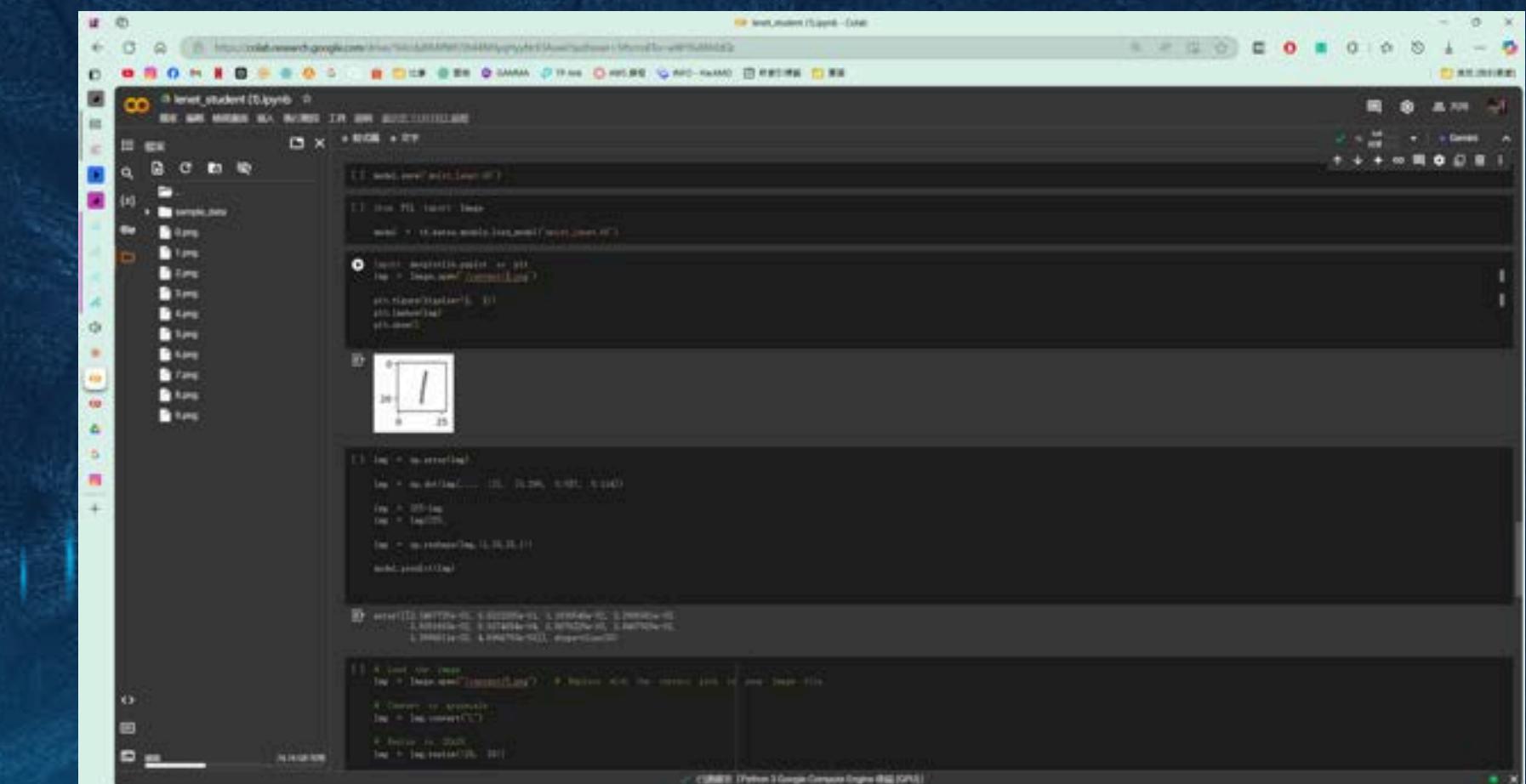
CNN

--LeNet LeNe

Yann LeCun

Iterations	ch Accuracy	100 epochs Accuracy
0	0.75	0.85
25	0.80	0.88
50	0.85	0.90
75	0.90	0.91
100	0.95	0.92

1. accurac
2. loss



02

PC

1.

PC

LeNet

1.

TensorFlow

MNIST

•

[0, 1]

$28 \times 28 \times 128 \backslash \text{times} 28 \backslash \text{times}$

$128 \times 28 \times 1$

One-Hot

2.

LeNet

$28 \times 28 \times 128 \backslash \text{times} 28 \backslash \text{times}$

•
 $128 \times 28 \times 1$

•

6 $5 \times 5 \times 1 \backslash \text{times} 5 \times 5$

○
2 $\times 22 \backslash \text{times} 22 \times 2$
16 $5 \times 5 \times 1 \backslash \text{times} 5 \times 5$

○
2 $\times 22 \backslash \text{times} 22 \times 2$

○
120
84
10

10

1.

3.

- 0.01

- crossentropy
-
-

accuracy

batch size 2048

epochs 100

steps per

epoch 30

SGD

categorical

4.

```
for i in range(100):
    train_step = sess.run(optimizer, feed_dict={x: batch_xs, y: batch_ys, keep_prob: 0.5})
    if i % 10 == 0:
        train_accuracy = accuracy.eval(feed_dict={x: batch_xs, y: batch_ys, keep_prob: 1.0})
        train_loss = loss.eval(feed_dict={x: batch_xs, y: batch_ys, keep_prob: 1.0})
        val_accuracy = accuracy.eval(feed_dict={x: validation_xs, y: validation_ys, keep_prob: 1.0})
        val_loss = loss.eval(feed_dict={x: validation_xs, y: validation_ys, keep_prob: 1.0})
        print("Epoch %d/%d" % (i+1, 100), "train step", i, "loss", train_loss, "acc", train_accuracy, "val loss", val_loss, "val acc", val_accuracy)
```

2.

(1)

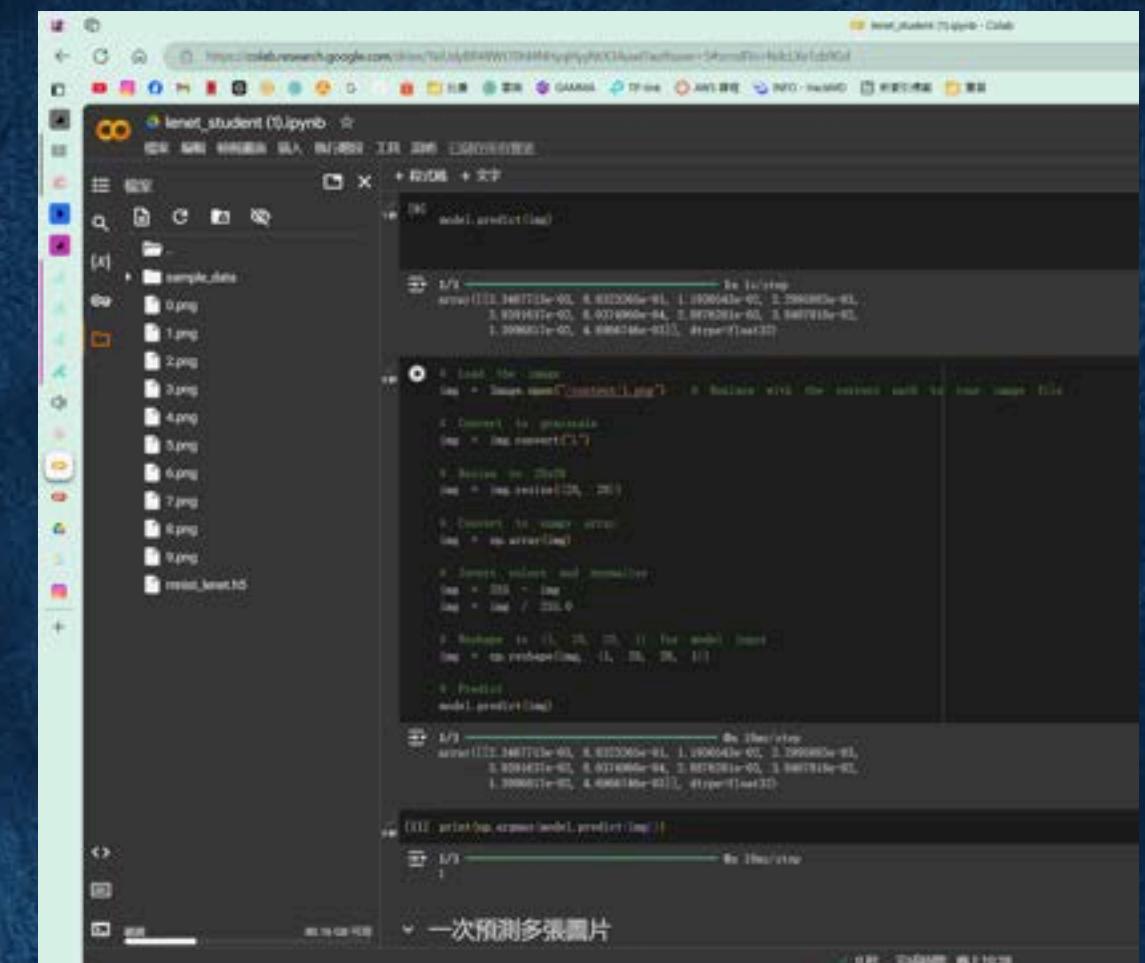
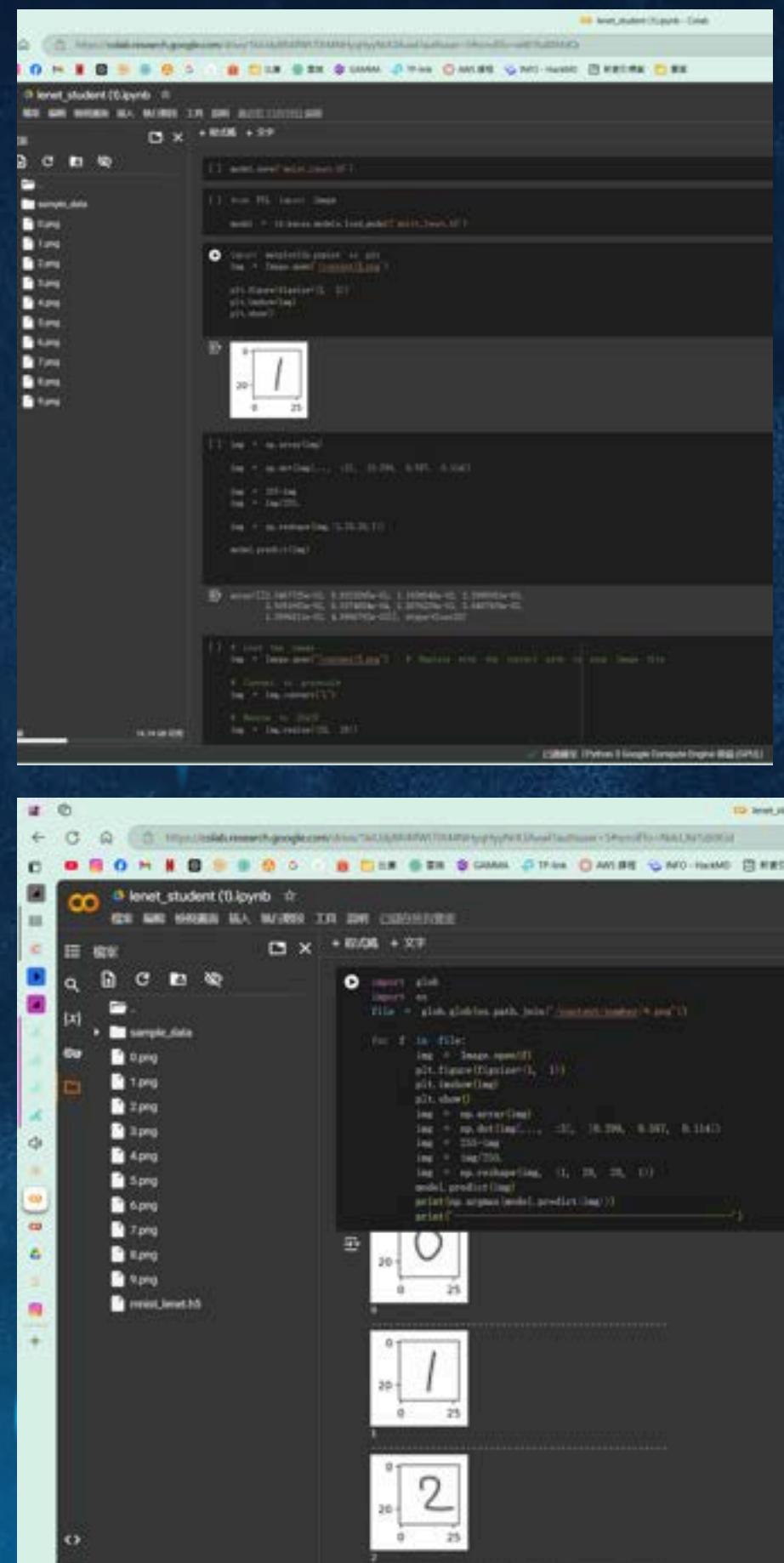
- 1

$28 \times 28 \times 128 \times 28 \times 128 \times 28 \times 1$

(2)

- 1

x_test



3.

3.

(1)

-

Matplotlib

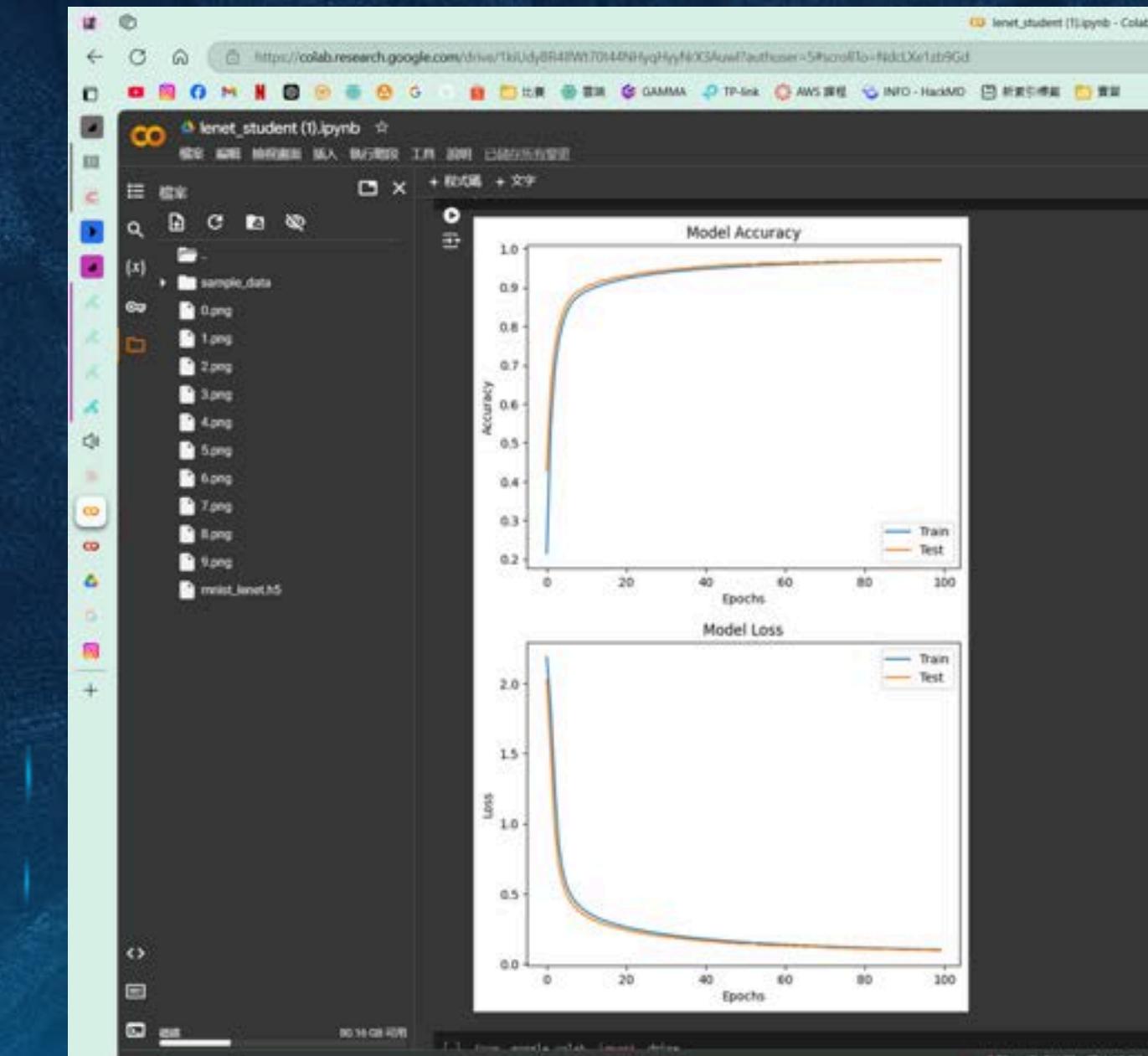
accuracy

loss

(2)

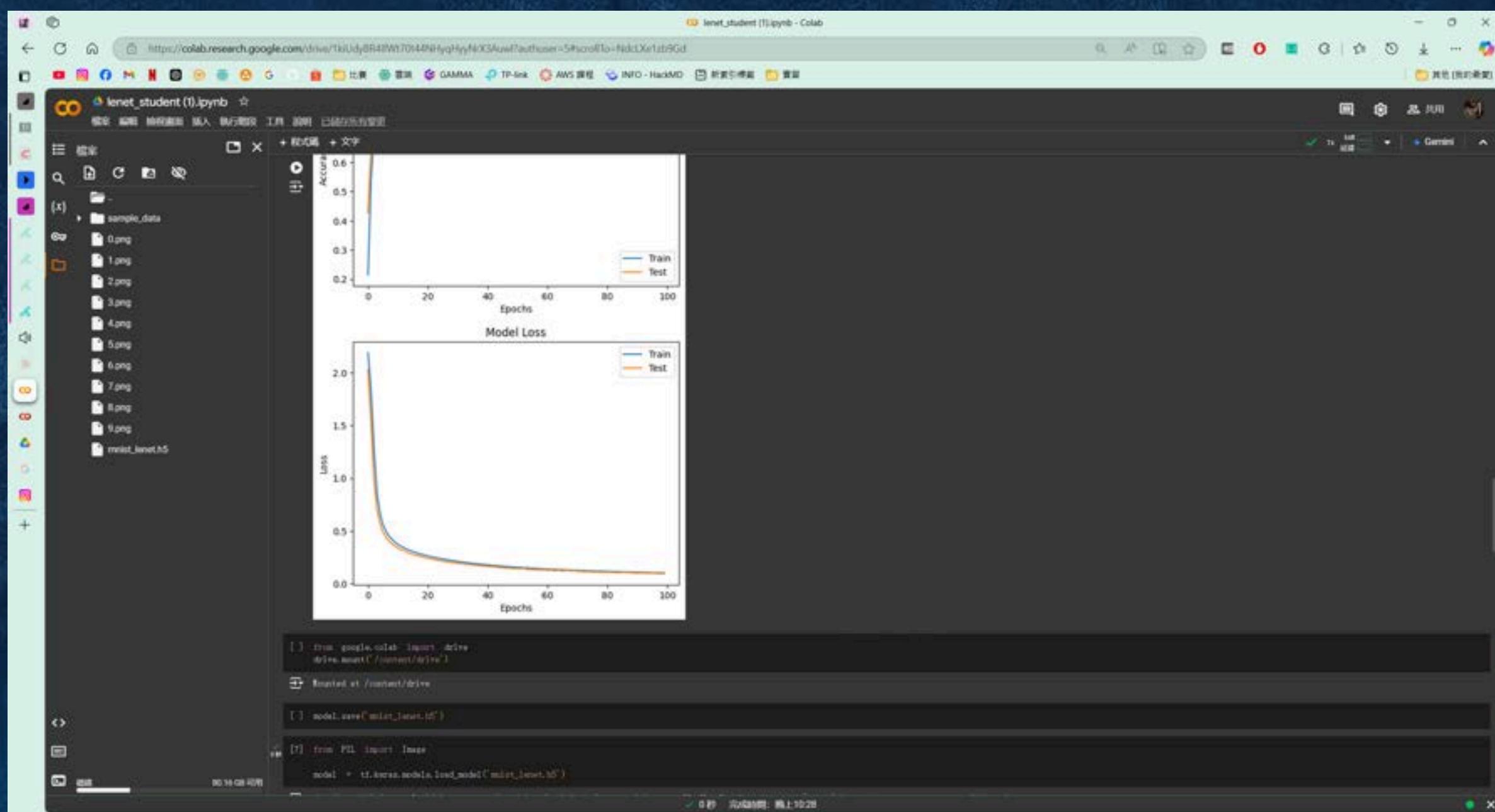
-

-



4.

HDF5



03

EDGE

1.

H5

H5 -> tflite

Dev board

tflite

mnist_tfite_test.ipynb

```
File Edit View Insert Cell Kernel Help Home 已运行的作业
```

搜索 目录 新建 + 代码块 + 文本

样本数据
分类图像.py
edgep1.ipynb
h5_to_image.py
label.txt
mnist_lenet.h5
mnist_tfite
mnist_tfite_test.ipynb

```
File Edit View Insert Cell Kernel Help Home 已运行的作业
```

自动保存: 1 秒钟
mnist_tfite.ipynb

自动保存由 Colaboratory
原文件位于
https://colab.research.google.com/drive/1BqG8tWu0U7v6oGtaXN7CJn7w

```
import tensorflow as tf
import numpy as np
import glob
import os
import matplotlib.pyplot as plt
from PIL import Image

print('TensorFlow 版本:', tf.__version__)

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

# 加载 CIFAR 数据集
(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, x_test)
x_train = np.reshape(
    x_train, (x_train.shape[0], x_train.shape[1], x_train.shape[2], 1))

def representative_data_gen():
    dataset_index = np.random.randint(0, 50000, 100)
    for i in range(100):
        image = x_train[dataset_index[i]]
        image = tf.image.resize(image, (IMAGE_SIZE, IMAGE_SIZE))
        image = tf.cast(image, tf.float32)
        image = tf.expand_dims(image, 0)
        yield image
```

The screenshot shows a Google Colab notebook titled "mnist_tflite_test.ipynb". The code cell contains Python code for loading a Keras model and converting it to TFLite format. The code includes comments explaining each step: loading the dataset, checking if the model file exists, loading the model, creating a converter, setting optimization parameters, and finally writing the TFLite model to a file. A warning message from TensorFlow is visible at the bottom of the cell.

```
dataset_file_index = np.random.choice(len(x_train), 100)
for i in range(100):
    image = x_train[dataset_file_index[i]]
    image = tf.image.resize(image, [IMAGE_SIZE, IMAGE_SIZE])
    image = tf.cast(image, tf.float32)
    image = tf.expand_dims(image, 0)
    print(image)

# 檢查模型文件是否存在
model_path = "/content/mnist_layer.h5"
if not os.path.exists(model_path):
    print("找不到模型文件: ", model_path)
    # 也可以在此添加其他載入模型的其他處理方式
    # 例如:
    # raise FileNotFoundError("模型文件不存在: " + model_path)
else:
    print("模型文件存在於: ", model_path)

# 加載模型
model = tf.keras.models.load_model(model_path)
print("模型已成功載入。")

# 轉換 .h5 到 .tflite
converter = tf.lite.TFLiteConverter.from_keras_model(model)

# 配置轉換器
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
converter.representative_dataset = representative_data_gen
converter.inference_input_type = tf.int8
converter.inference_output_type = tf.int8

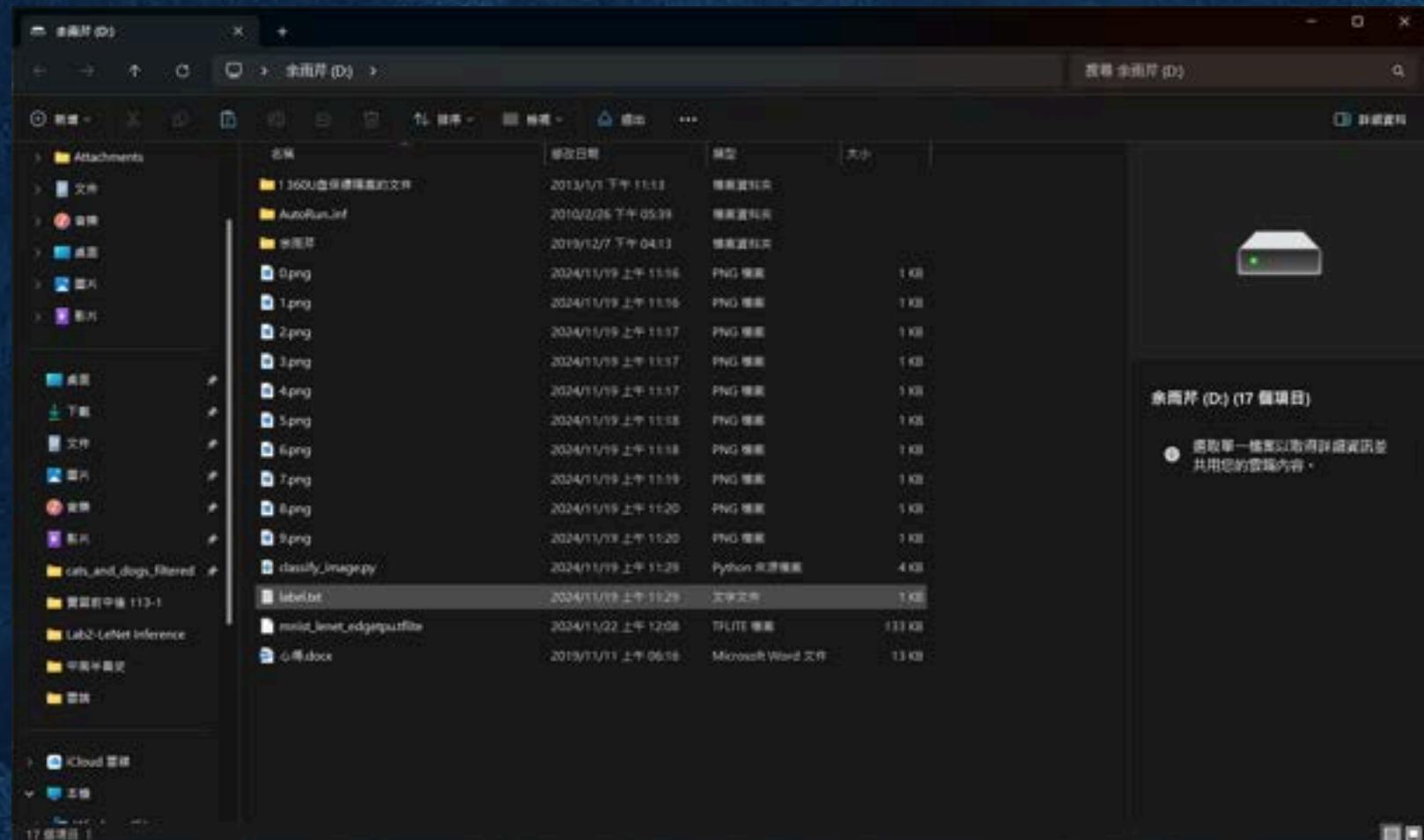
# 轉換模型
try:
    tflite_model = converter.convert()
    print("模型已成功轉換為 TFLite 標式。")
except Exception as e:
    print("轉換模型時發生錯誤: ", e)
    exit(0)

# 寫出 TFLite 標型
tflite_model_path = '/content/mnist_layer.tflite'
with open(tflite_model_path, 'wb') as f:
    f.write(tflite_model)
print("TFLite 標型已存放在: ", tflite_model_path)
```

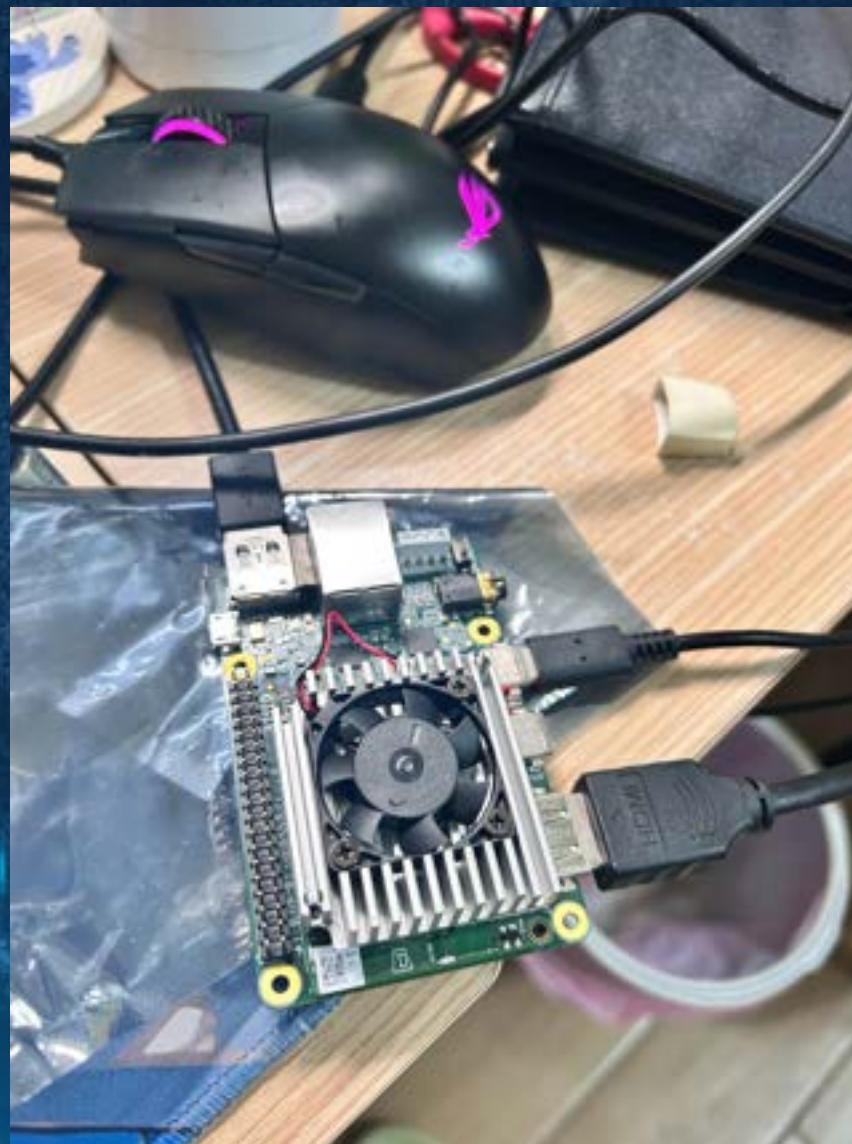
TensorFlow 版本: 2.17.1
/usr/local/lib/python3.10/dist-packages/tensorflow/python/training/optimizer/base_optimizer.py:33: UserWarning: Argument 'decay' is no longer supported and will be ignored.
warnings.warn()
WARNING:root:Compiled the loaded model, but the compiled metrics have yet to be built. "model.compile_metrics" will be empty until you train or evaluate the model.
模型文件存在的位置: /content/mnist_layer.h5
模型已成功載入。
Saved artifact at '/tmp/tmhpjxuW'. The following endpoints are available:
* Endpoint 'serve'
 axes=3 (OPTIONAL, INT32) TensorSpec(shape=[None, 28, 28, 1], dtype=tf.float32, name='input_2')
 Output Type:
 TensorSpec(shape=[None, 10], dtype=tf.int32, name='Real')
 Captures:
 128874425707070: TensorSpec(shape=(), dtype=tf.resource, name='flow')
 128874425706468: TensorSpec(shape=(), dtype=tf.resource, name='flow')
 128874425705716: TensorSpec(shape=(), dtype=tf.resource, name='flow')
 128874425705487: TensorSpec(shape=(), dtype=tf.resource, name='flow')
 128874425742016: TensorSpec(shape=(), dtype=tf.resource, name='flow')
 128874425757088: TensorSpec(shape=(), dtype=tf.resource, name='flow')
 128874425752281: TensorSpec(shape=(), dtype=tf.resource, name='flow')
 12887442572144: TensorSpec(shape=(), dtype=tf.resource, name='flow')
 12887442572136: TensorSpec(shape=(), dtype=tf.resource, name='flow')
 128874425772705: TensorSpec(shape=(), dtype=tf.resource, name='flow')
模型已成功轉換為 TFLite 標式。
TFLite 標型文件存在於: /content/mnist_layer.tflite
/usr/local/lib/python3.10/dist-packages/tensorflow/lite/python/converter.py:93: UserWarning: Statistics for quantized inputs were expected, but not specified; continuing anyway.
warnings.warn()

2. Dev board

1. 將需要的檔案放入USB

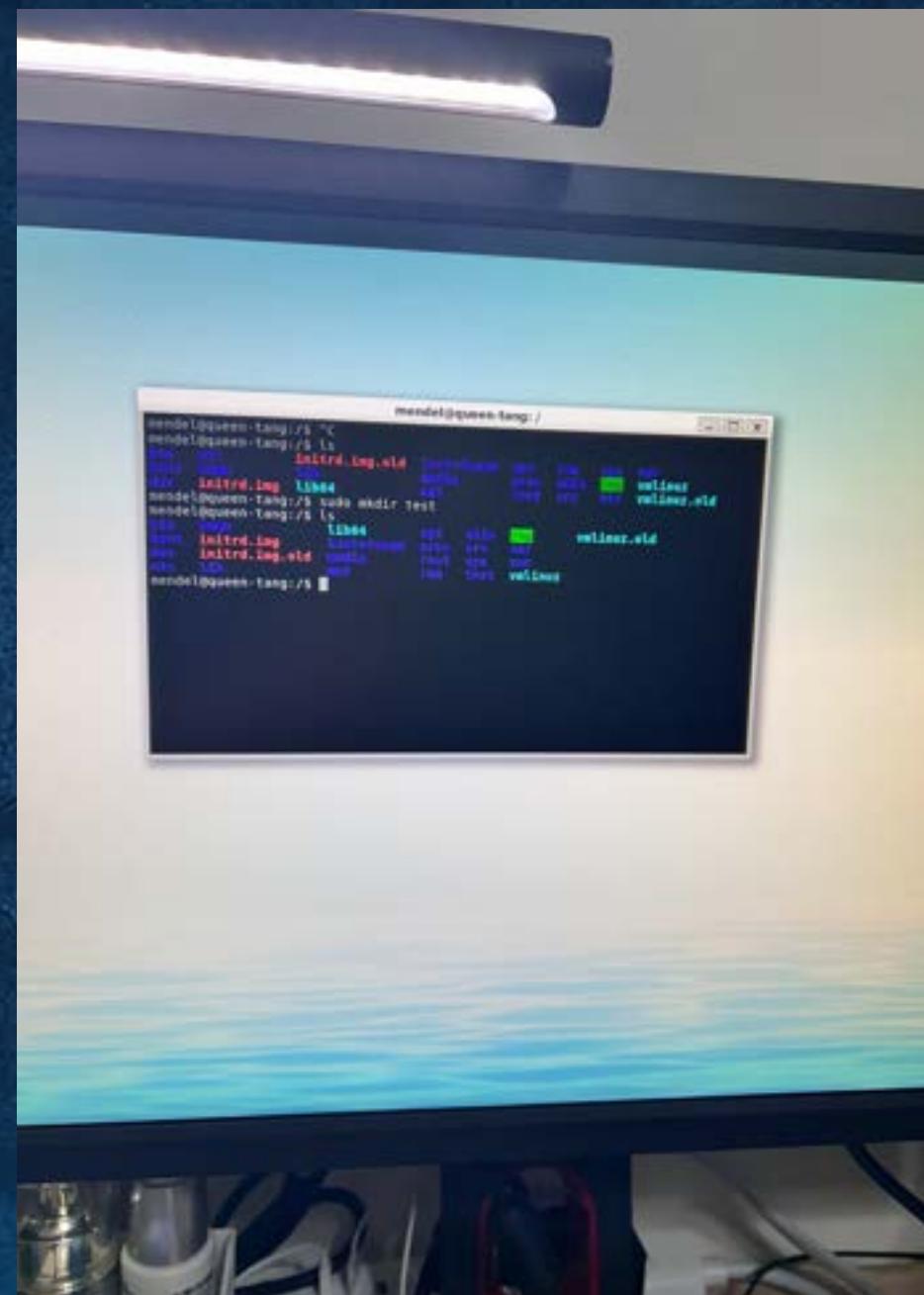


2. 將Dev board接上螢幕、滑鼠和鍵盤



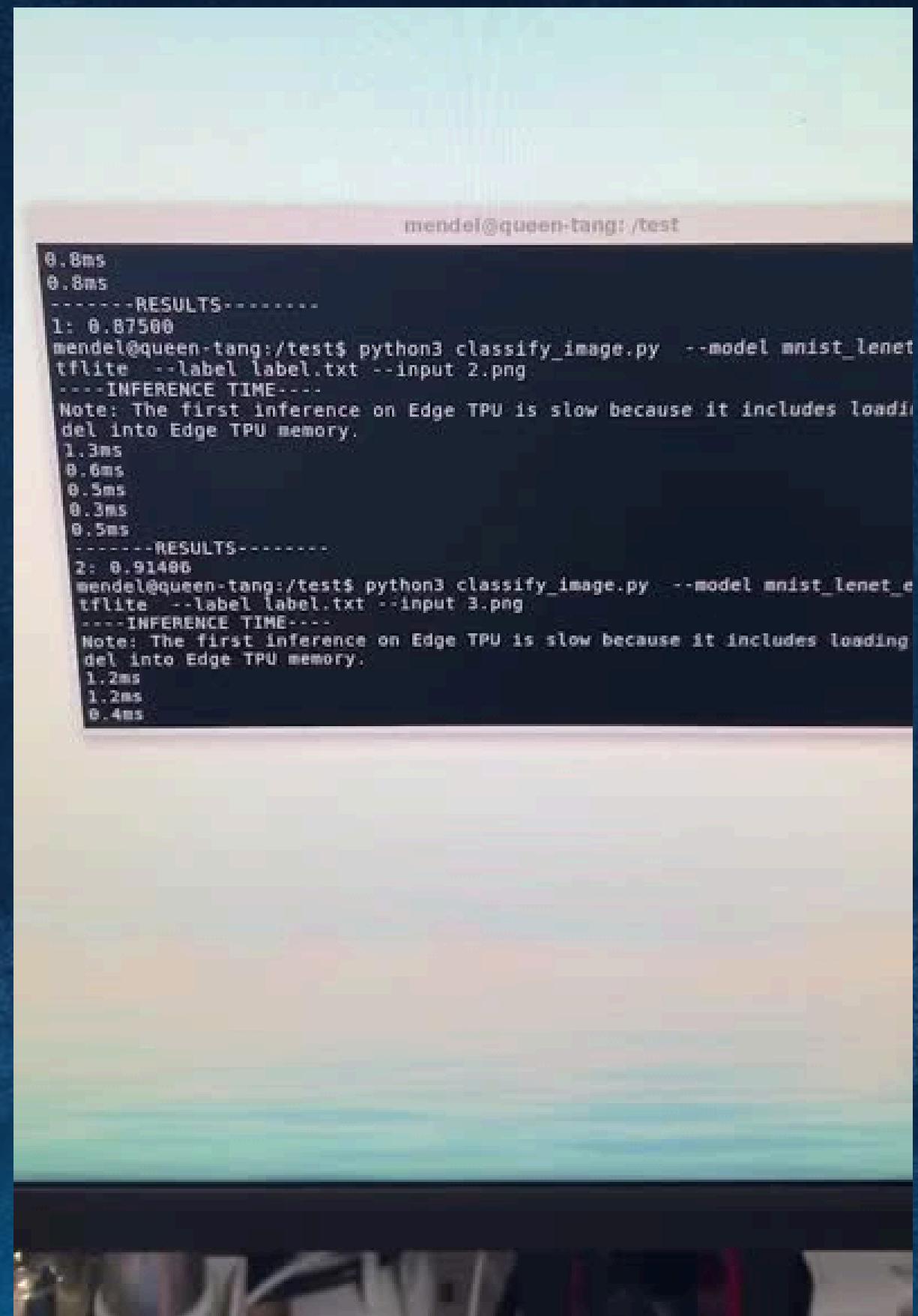
2. Dev board

3. 創資料夾掛載USB



2. Dev board

4. 輸入指令



```
mendel@queen-tang:/test$ python3 classify_image.py --model mnist_lenet_tflite --label label.txt --input 2.png
-----RESULTS-----
1: 0.87500
mendel@queen-tang:/test$ python3 classify_image.py --model mnist_lenet_edgetpu --label label.txt --input 3.png
-----RESULTS-----
2: 0.91486
Note: The first inference on Edge TPU is slow because it includes loading model into Edge TPU memory.
```

The terminal window displays the results of two image classification runs. The first run, using the TensorFlow Lite model, has a result of 0.87500. The second run, using the Edge TPU model, has a result of 0.91486. A note indicates that the first inference on Edge TPU is slow due to model loading.

影片觀看請點我

04

EDGE

```
mendel@queen-tang:/test  
0.8ms  
0.8ms  
-----RESULTS-----  
1: 0.87500  
mendel@queen-tang:/test$ python3 classify_image.py --model mnist_lenet_tflite --label label.txt --input 2.png  
---INFERENCE TIME---  
Note: The first inference on Edge TPU is slow because it includes loading model into Edge TPU memory.  
1.3ms  
0.6ms  
0.5ms  
0.3ms  
0.5ms  
-----RESULTS-----  
2: 0.91486  
mendel@queen-tang:/test$ python3 classify_image.py --model mnist_lenet_edtflite --label label.txt --input 3.png  
---INFERENCE TIME---  
Note: The first inference on Edge TPU is slow because it includes loading model into Edge TPU memory.  
1.2ms  
1.2ms  
0.4ms
```

05

問題 1

Protobuf TensorFlow

TensorFlow Protobuf

TensorFlow

Protobuf 3.21.0

Protobuf 3.20.x

Protobuf

解決問題

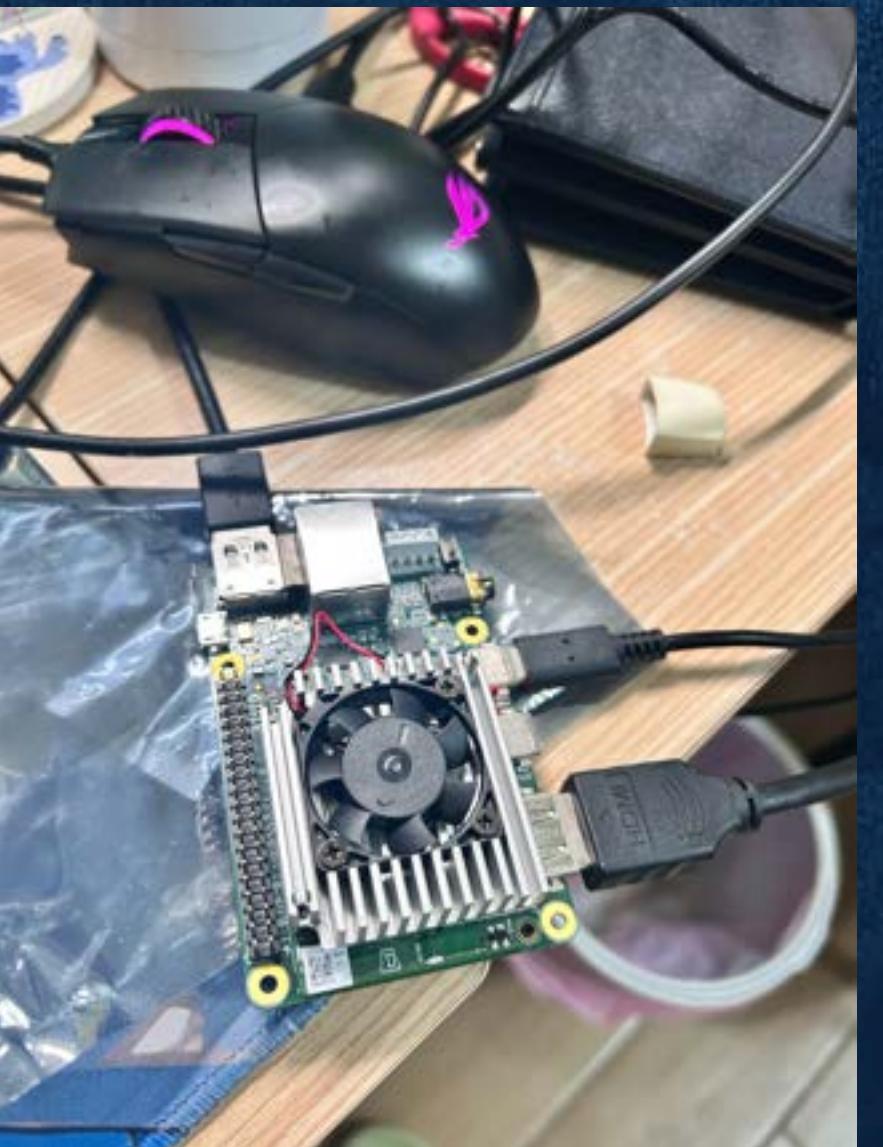
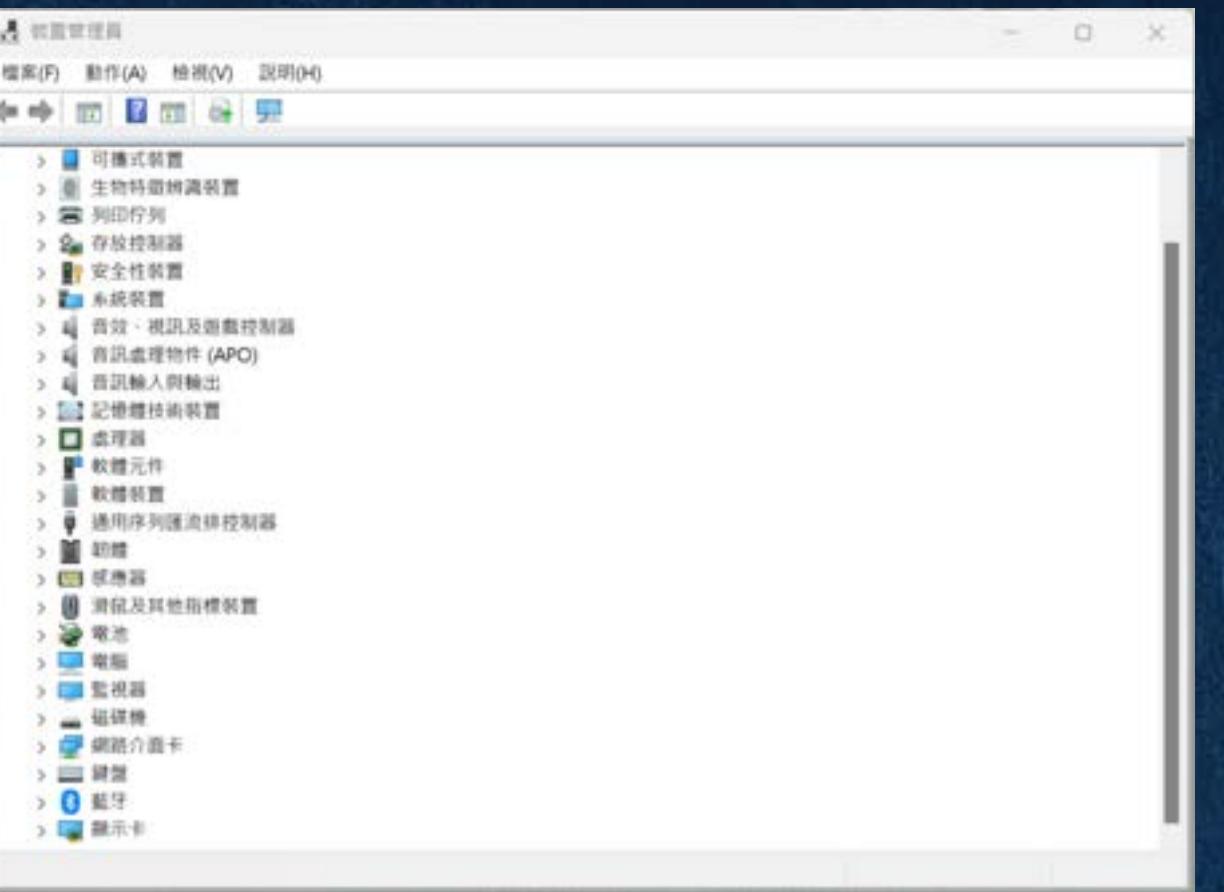
核心在於 控制 TensorFlow 與 Protobuf 的版本，避免版本不匹配
導致的錯誤：

1. 卸載不相容版本：避免舊版本殘留。
 2. 指定版本重新安裝：選擇相容的組合，安裝了特定的 TensorFlow 版本 (2.8.0) 和 Protobuf 版本 (3.20.0)。
 3. 重啟環境：確保所有改動正確應用。

問題 2

OTG

解決問題
直接在 Dev board 上接上螢幕、滑鼠和鍵盤



06

心得

在這次 Lab-2 的實作過程中，我學習並完成了從 LeNet 模型的設計、訓練、到部署的完整流程，並針對 PC 端與 Edge 端的環境進行了配置與測試。以下是在這次實驗中的體會與收穫：

學習與挑戰

- 深度學習模型的構建與訓練：在 PC 端，我成功利用 TensorFlow 實現了 LeNet 模型的構建，並在 MNIST 資料集上進行了訓練與推理。過程中，學習到如何優化模型的結構和參數設定，以及如何觀察模型的準確率與損失曲線來調整訓練策略。
- 模型的轉換與部署：訓練完成後，我進一步將 H5 格式的模型轉換為適合開發板使用的 tflite 格式，並使用全整數量化技術以適配資源有限的 Edge 端環境。這一部分使我更深刻地理解了模型壓縮與優化的重要性。
- 硬體配置與環境搭建：在實現 USB OTG 連接時，曾遇到開發板與電腦無法正確識別的問題，經過檢查驅動與硬體連接，最終成功配置了螢幕、滑鼠和鍵盤，並構建出完整的開發環境。
- 遇到問題與解決：過程中，我遇到了多次技術挑戰，包括 TensorFlow 與 Protobuf 版本不相容、USB OTG 的供電與設備識別問題等。在這些過程中，我學會了如何快速查找問題根源並有效解決，增強了對開發工具和系統環境的掌控力。

收穫與未來展望

- 這次實驗不僅讓我更全面地了解了 LeNet 模型的結構與應用，也讓我對深度學習在嵌入式設備中的實現流程有了更深刻的認識。我體會到，理論與實踐相結合的重要性在於，實際操作過程中的每一個細節都會影響到最終的實驗效果。

THANK YOU!

第四組-四資管AI三A-B11123224-余雨芹