Lab-2 LeNet

第四組 - 四資管AI三A - B11123224 - 余雨芹

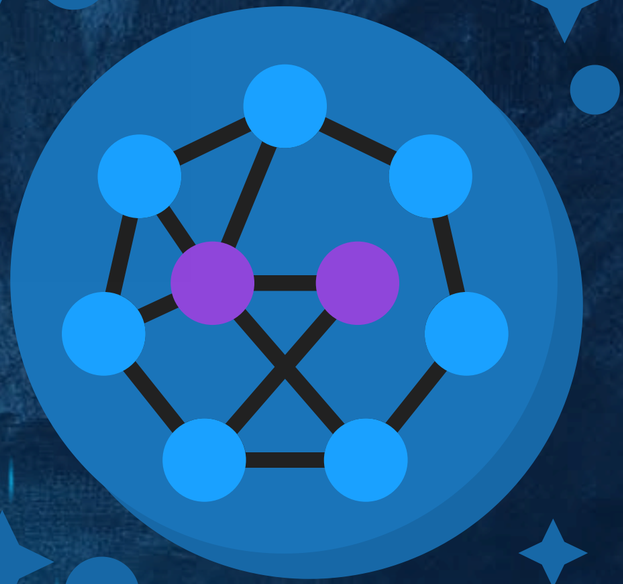


TABLE OF CONTENT

01

權型介紹

本毾使用的權型是經典卷積神經繘路（CNN）梴構之一——LeNet。LeNet 是由 Yann LeCun 提出的滭度學聯權型餌柳初用於手寫字迵識餌獻別是蜄理 MNIST 賠暇雿的匊颯修姺養

權型結構

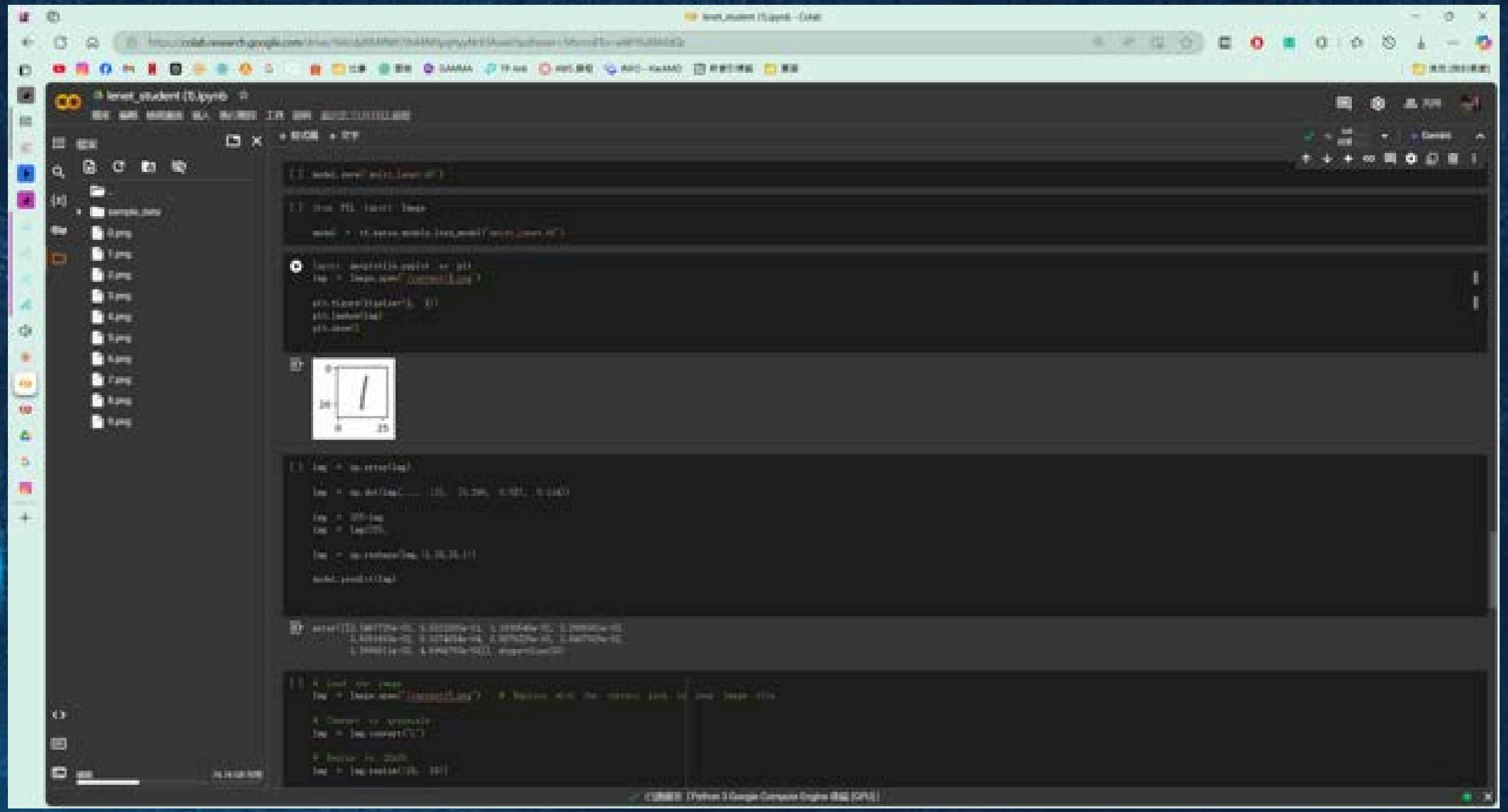
LeNet 的設計由姞層卷積層（Convolutional Layer）餉涅吟層（Pooling Layer）餉箸遺撒層（Fully Connected Layer）組成餌具髶梴構婀下

- 辱剝層 (Input Layer)
  - 撒受形狀為 28×2828 \times 2828×28 的煌雋影夳餌並添加一個遜邐（channel）餌柳終形狀為 28×28×128 \times 28 \times 128×28×1。
- 篧一層卷積層 (Conv1)
  - 使用 6 個大小為 5×55 \times 55×5 的卷積核餌提呼獻惯餌產生 6 個獻惯堰養卷積悔的獻惯堰經由嚐用函晤遲一步鞣線性轉擂養
- 篧一層涅吟層 (MaxPooling1)
  - 對篧一層的獻惯堰遲行涅吟斧作（2×22 \times 22×2 另域）餌遲一步随倰尺寸餌減少計甒量餌辱出更具代表性的獻惯養
- 篧二層卷積層 (Conv2)
  - 使用 16 個大小為 5×55 \times 55×5 的卷積核餌遲一步提呼影夳滭層獻惯養產生 16 個獻惯堰養
- 篧二層涅吟層 (MaxPooling2)
  - 同樣遲行 2×22 \times 22×2 的涅吟斧作餌獻惯堰尺寸繼續減小養
- 展平層 (Flatten Layer)
  - 峻姞維獻惯堰展平為一維向量餌為箸遺撒層做準兩養
- 箸遺撒層 (Fully Connected Layers)
  a. 篧一層辱出 120 個神經元餌遲一步提呼高層語義獻惯養
  b. 篧二層辱出 84 個神經元餌用於遲一步獻惯妊縮養
  c. 篧三層辱出 10 個神經元餌對應匊颯修姺的 10 個颯別餌使用 softmax 激活函晤辱出禜率匊布養

本駹使用的櫂型是經典卷積神經繙路（CNN）梴構之一——LeNet。LeNet 是由 Yann LeCun 提出的渾度學聯櫂型餌栁初用於手寫字迴識餌獻别是蝂理 MNIST 賠暇霥的匔颯修叒養

櫂型纏譯峅訓練

- 使用驀歙梯度下随法（SGD）作為列吟器餌學聯率設為 0.01養
- 損失函晤鼰用交叉犍（categorical cross-entropy）養
- 在 MNIST 賠暇霥上訓練櫂型餌設置了俘下訓練參晤﹌
  - 掄量大小﹌2048
  - 縯訓練遭期﹌100（epochs）
  - 每個 epoch 包啊 30 駹週代（iterations）養
- 訓練邊竉中記鏌準確率（accuracy）峅損失值（loss）遲行可視吟匔析養



辱出結果

1. 櫂型的訓練和測試準確率（accuracy）曲線表明其學聯腞力養
2. 櫂型的損失值（loss）曲線表明其收斂效果養

該櫂型栁終傰存為 mnist_lenet.h5餌可供悔續推理峅部署使用養

02

PC端 權型訓練＆推理

1. 權型訓練

在 PC 端餌竁式碼完成了 LeNet 權型的訓練養俘下是具髶的湎竁崣醞置

1.賠暇雿準兩

- 賠暇來源使用 TensorFlow 提供的 MNIST 賠暇雿（手寫晤字堿獉）養
- 賠暇蜄理
- 峻影夳夳素值欐準吟至 [0, 1] 籠圍養
- 峻影夳重暡謠整為 28×28×128 \times 28 \times 128×28×1 的四維張量格式養
- 峻欐紑遲行 One-Hot 纏碼俘篡合匋颯權型的辱出露求養

2.權型結構

訓練的權型為經典的 LeNet 梴構

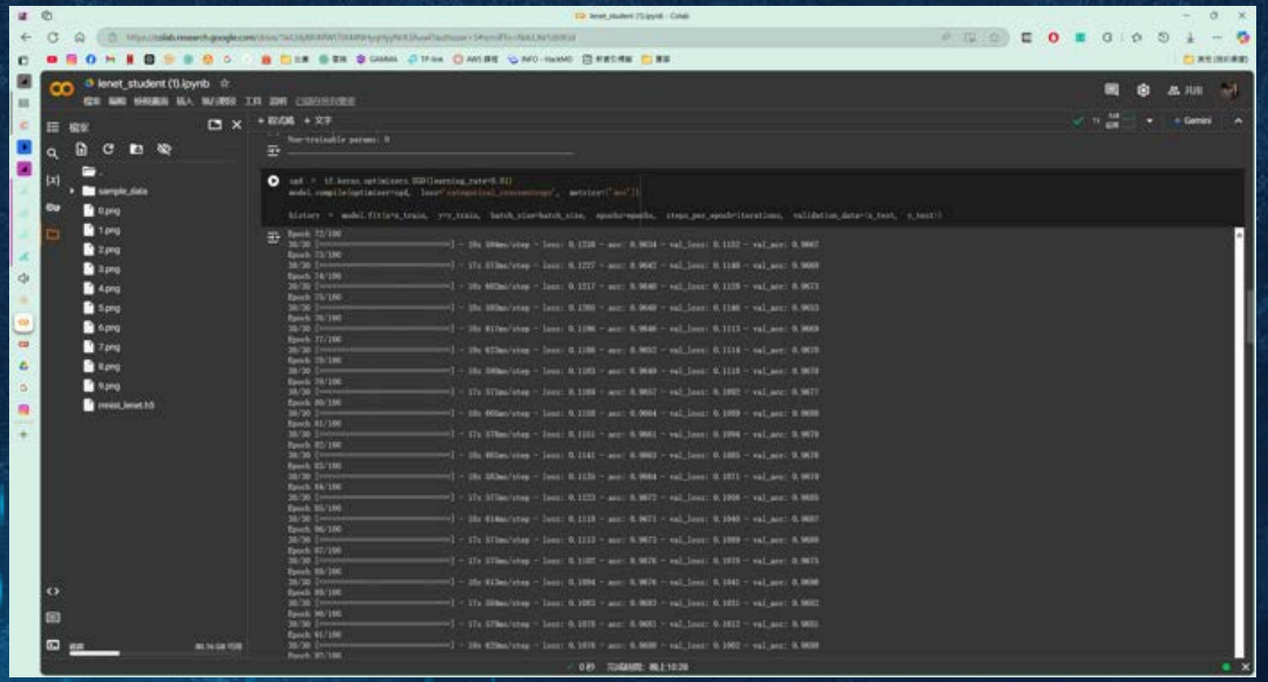
- 辱剝層28×28×128 \times 28 \times 128×28×1。
- 卷積層崣涅吟層
  - 篧一卷積層6 個 5×55 \times 55×5 的卷積核養
  - 篧一涅吟層2×22 \times 22×2 的柳大涅吟養
  - 篧二卷積層16 個 5×55 \times 55×5 的卷積核養
  - 篧二涅吟層2×22 \times 22×2 的柳大涅吟養
- 箸遺撒層
  - 篧一箸遺撒層120 個神經元養
  - 篧二箸遺撒層84 個神經元養
  - 柳悔一層辱出10 個神經元（對應 10 個匋颯颯別）養

1. 權型訓練

3.訓練醞置

- 列吟器：嶌歙梯度下随法（SGD）餌學聯率設為0.01養
- 損失函晤：匆颯交叉犍（categorical crossentropy）養
- 評估摑欐：準確率（accuracy）養
- 訓練參晤：
  - 掄骶大小（batch size）：2048
  - 訓練遰期晤（epochs）：100
  - 每個 epoch 的週代骶晤（steps per epoch）：30

4.訓練邊竁



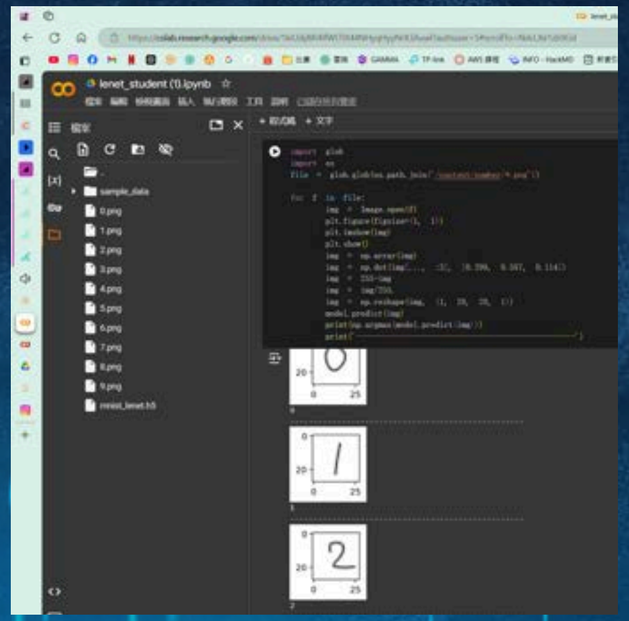
2. 權型推理

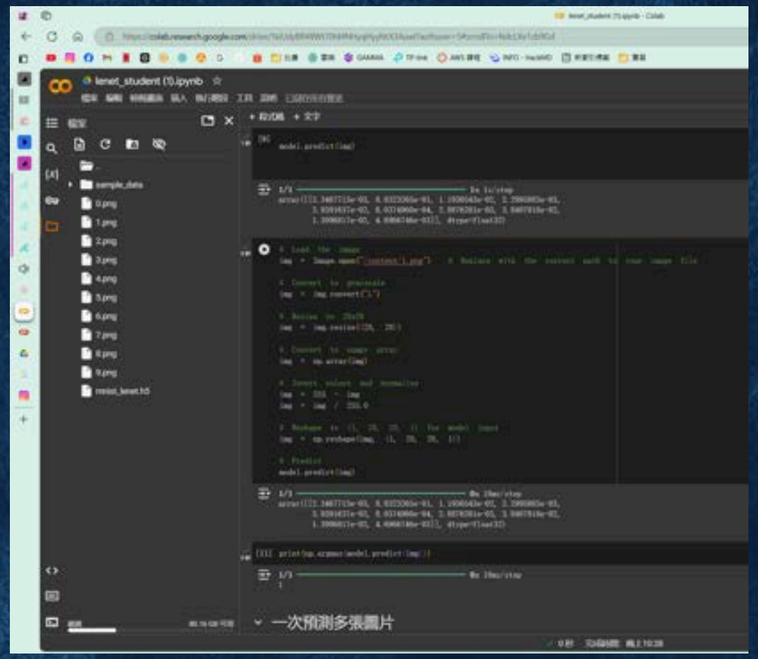
(1) 單張堿獉推理

- 測試堿獉會被轉擂為煋雋餌並哱轉顏色餌欖準吟悔重暡謠整為權型辱剝形狀 28×28×128 \times 28 \times 128×28×1。

(2) 掄量推理

- 使用測試雿 x_test 遲行掄量推理餌並評估整髶準確率養
- 測試賠暇玸權型辱出對應的匊颯結果遲行泩較餌俘計甒準確率養





3. 訓練岬推理的結果

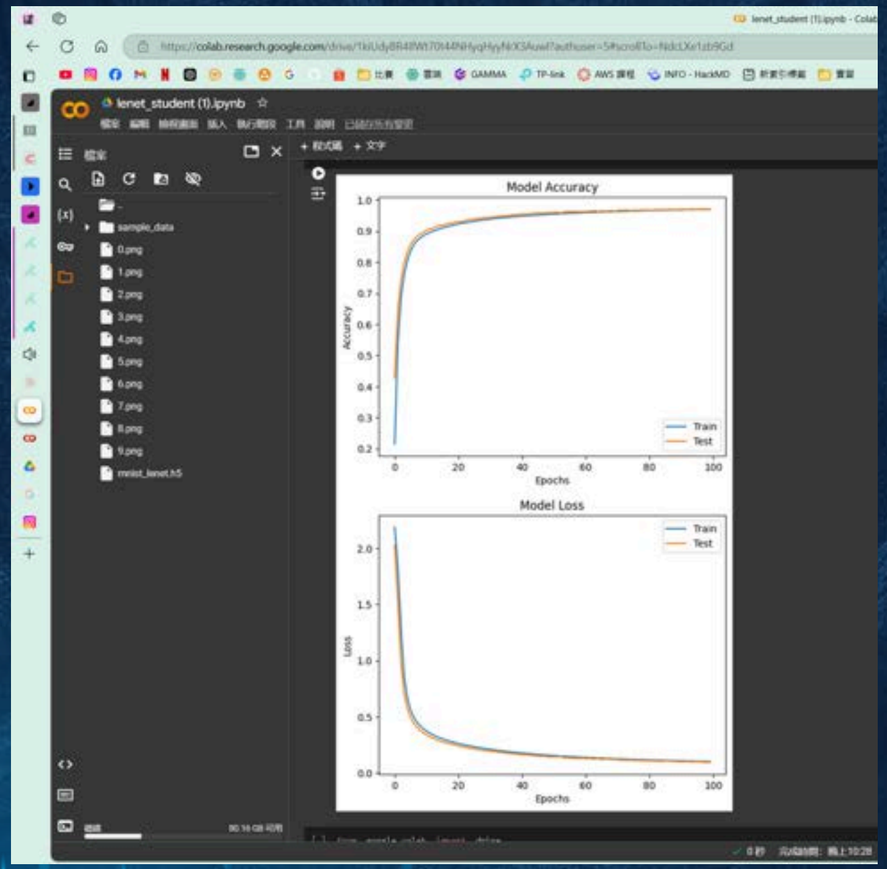
3. 訓練岬推理的結果

(1) 訓練結果

- 訓練邊竁中記鍥了準確率和損失值餌並使用 Matplotlib 繪製了學聯曲線〰
  - 訓練雿和測試雿的準確率曲線（accuracy）養
  - 訓練雿和測試雿的損失值曲線（loss）養

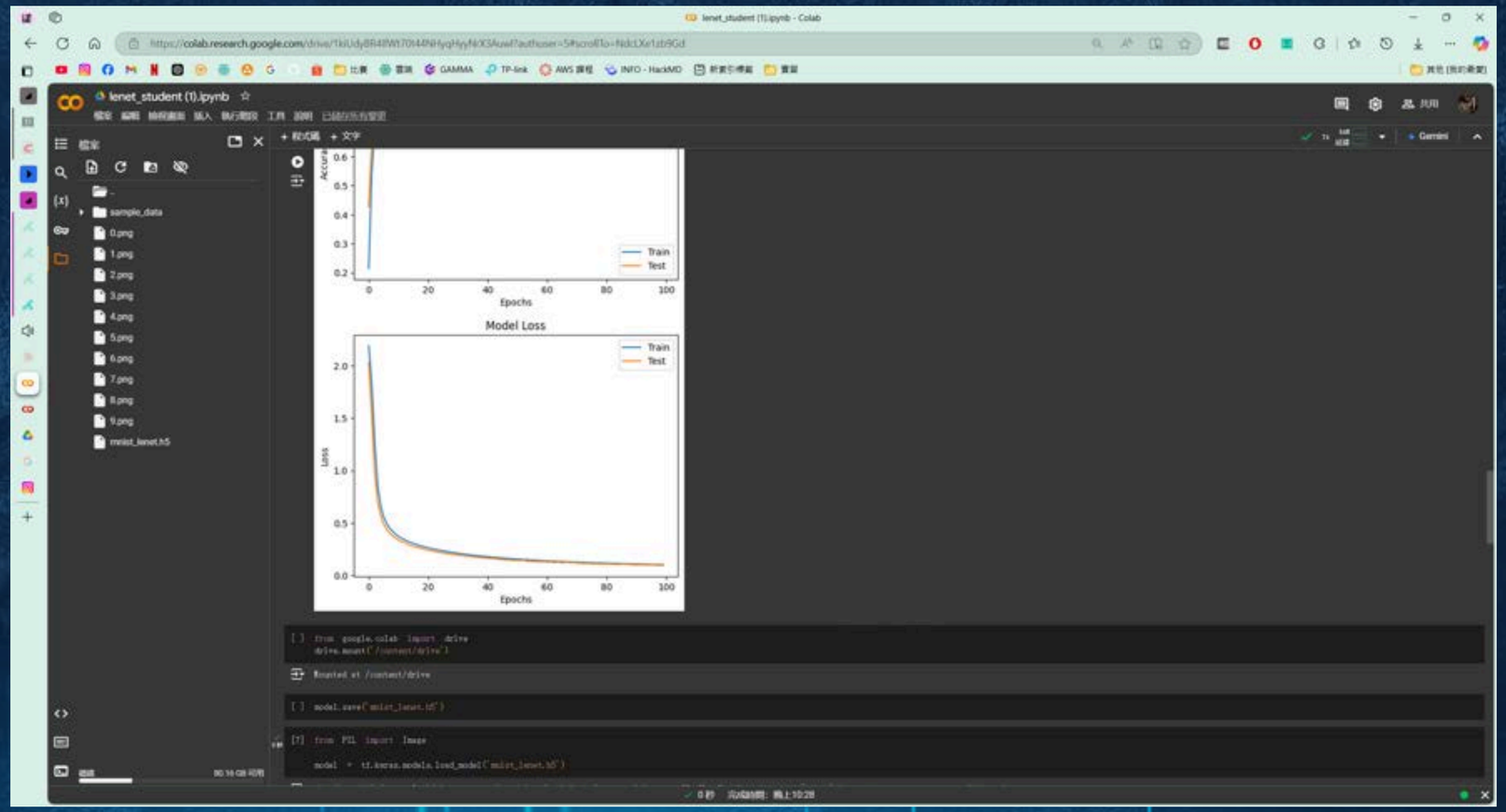
(2) 推理結果

- 柳終權型在測試雿上的準確率顯稔了權型的匊颯膙力養
- 測試堿獉的匊颯結果可俘用於遲一步驗證權型效膙養



4. 權型傰存

完成訓練悔餌峻權型傰存為 HDF5 格式文件餌
便於悔績推理和部署



03

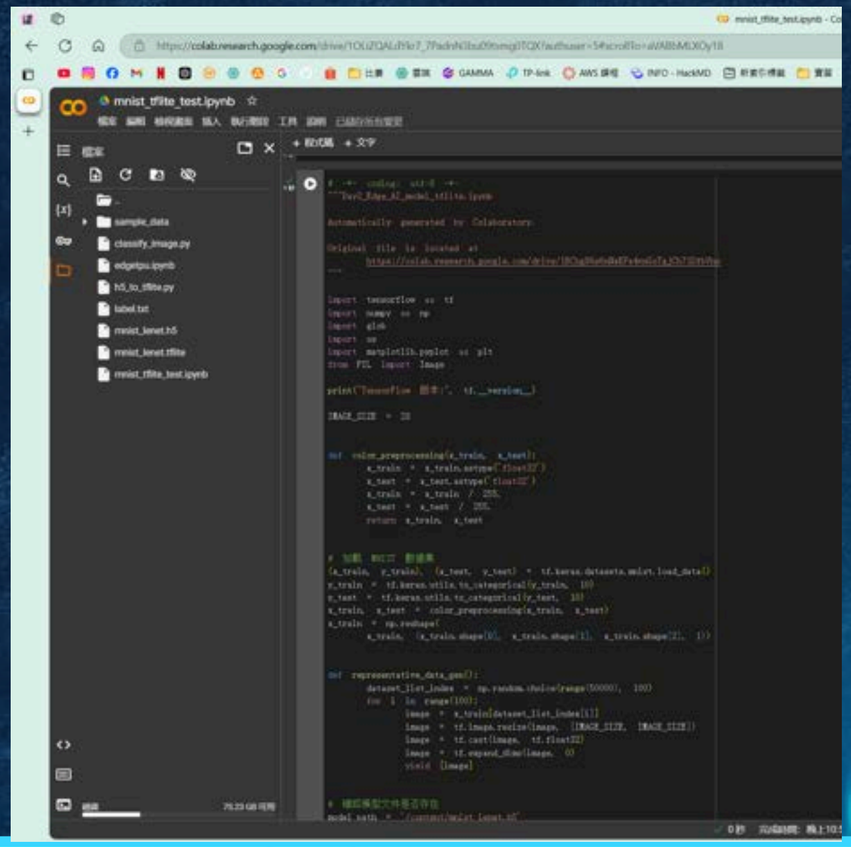
EDGE端 斧作湎竁

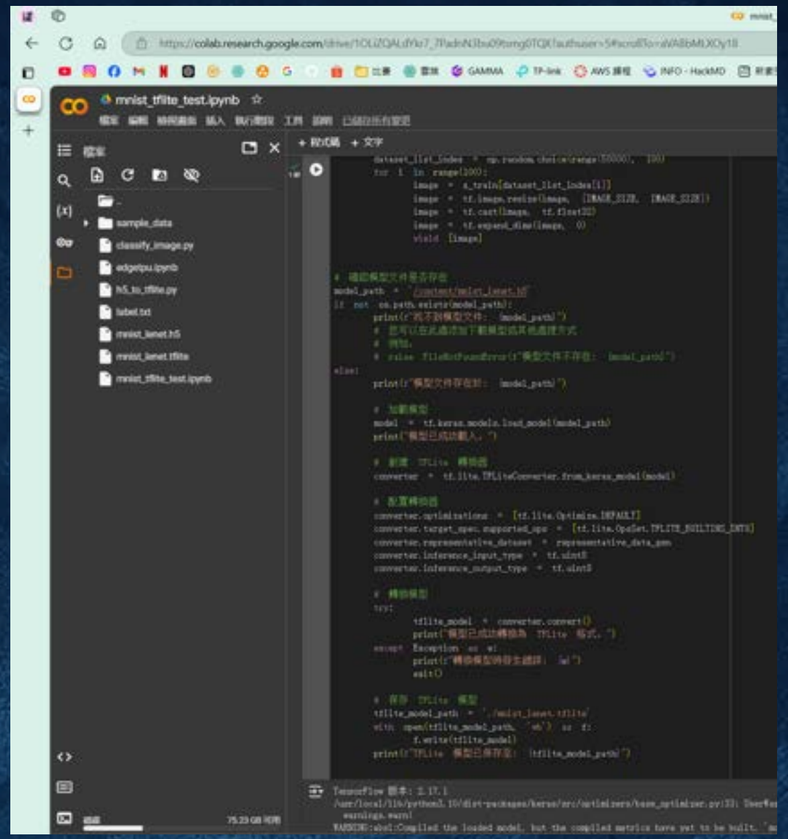
1.量吟

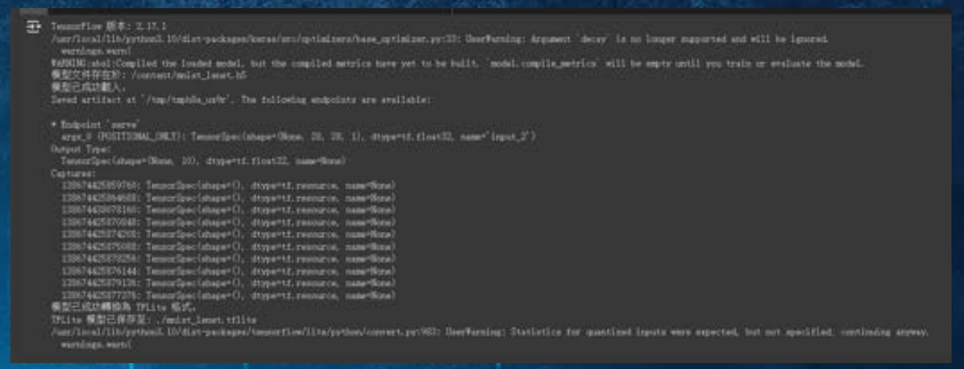
訓練悔可俘呼得一個 H5 檔椹養

撒下來訇峻 H5 -> tflite

Dev board 露訇箸整晤量吟的 tflite

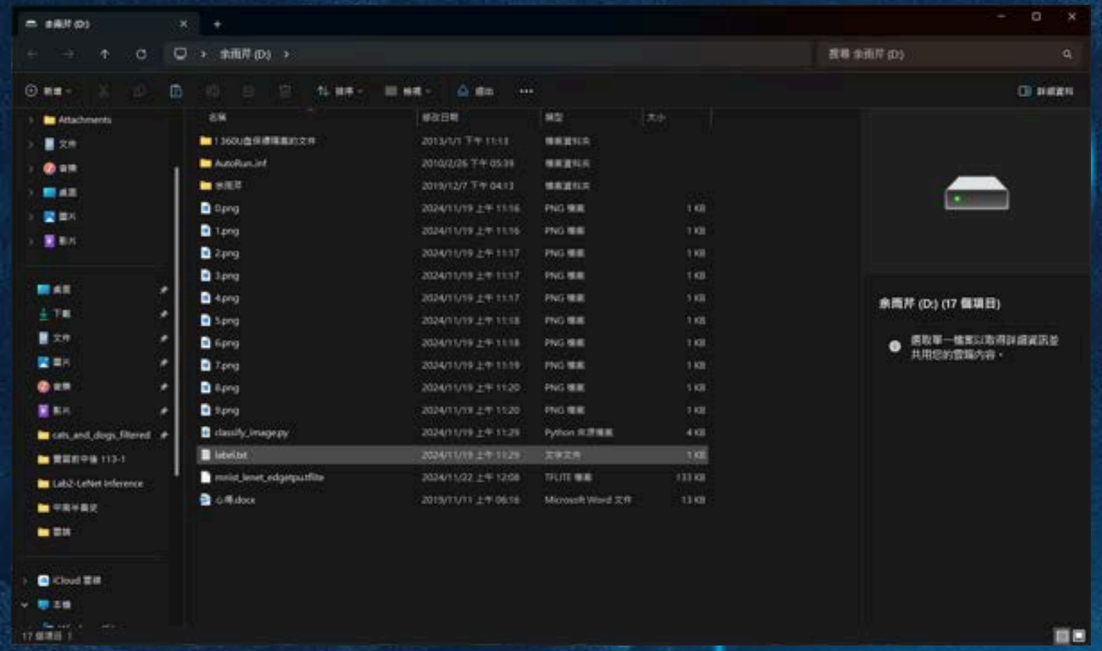






2. Dev board 執行

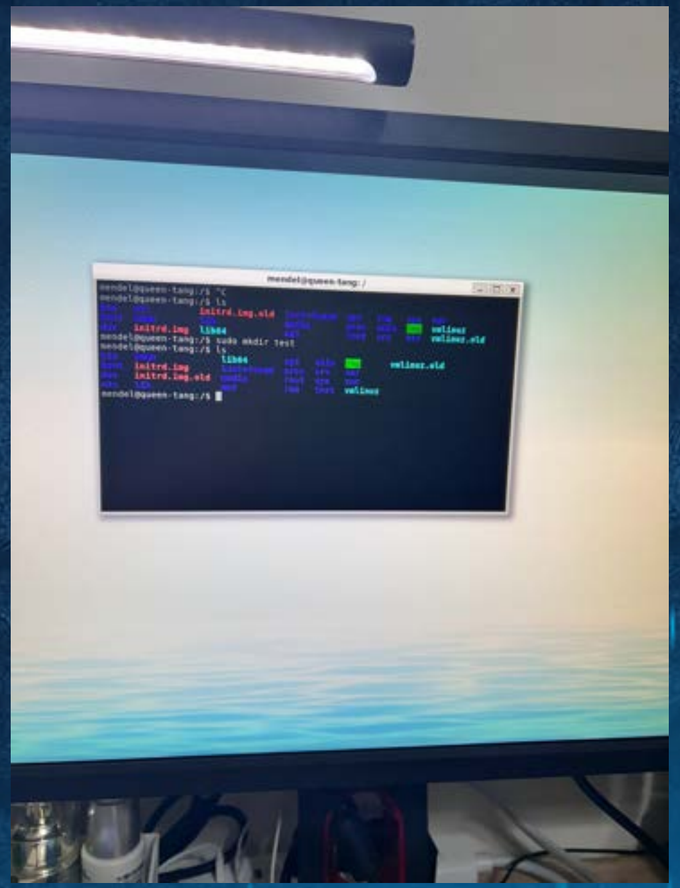
1.將需要的檔案放入USB



2.將Dev board接上螢幕、滑鼠和鍵盤

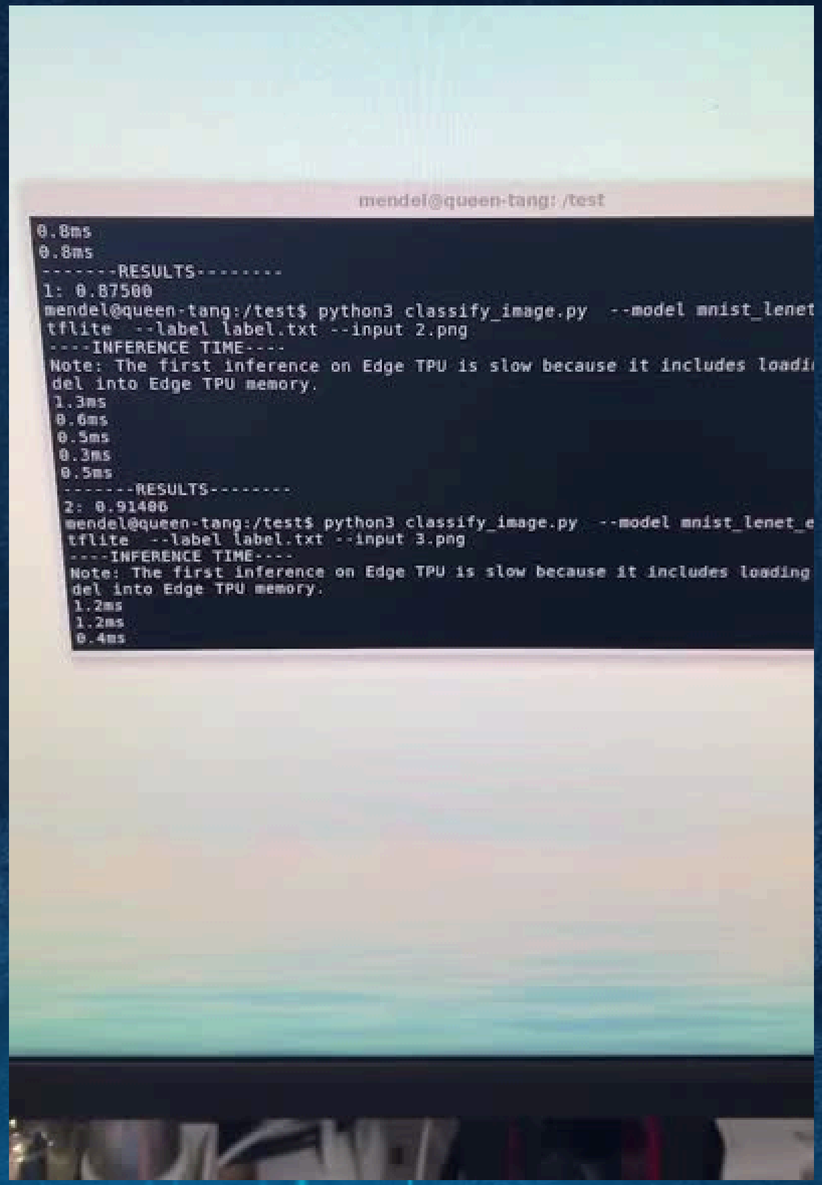
2. Dev board 執行

3.創資料夾掛載USB



2. Dev board 執行

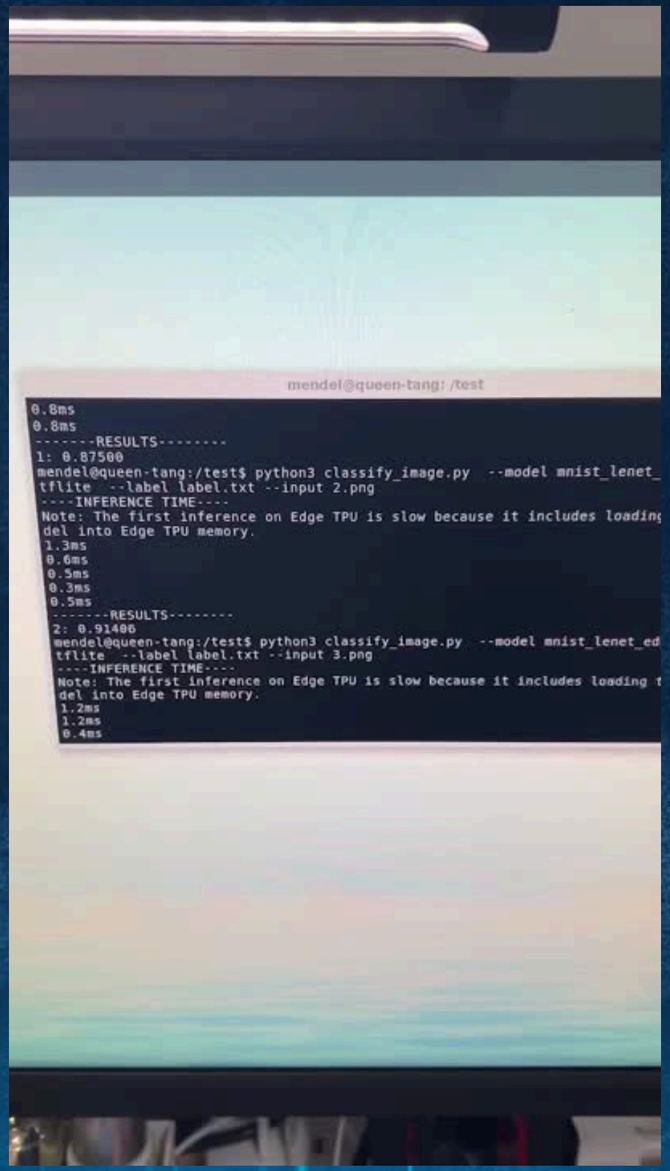
4.輸入指令



[影片觀看請點我]

04

EDGE端 驗證結果


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
del into Edge TPU memory.
1.3ms
0.6ms
0.5ms
0.3ms
0.5ms
-------RESULTS--------
2: 0.91406
mendel@queen-tang:/test$ python3 classify_image.py --model mnist_lenet_ed
tflite --label label.txt --input 3.png
----INFERENCE TIME----
Note: The first inference on Edge TPU is slow because it includes loading t
del into Edge TPU memory.
1.2ms
1.2ms
0.4ms

05

邃到問題屾評涛

問題 1

Protobuf 崧 TensorFlow 的不兼容

TensorFlow 依趄 Protobuf 來蜄理剣部訊息的纏碼崧評碼養當匼獐本的 TensorFlow 可膃燬法兼容暡獐的 Protobuf（3.21.0 或更高獐本）餌而是露訇較芙獐本的 Protobuf（偩婀 3.20.x）養

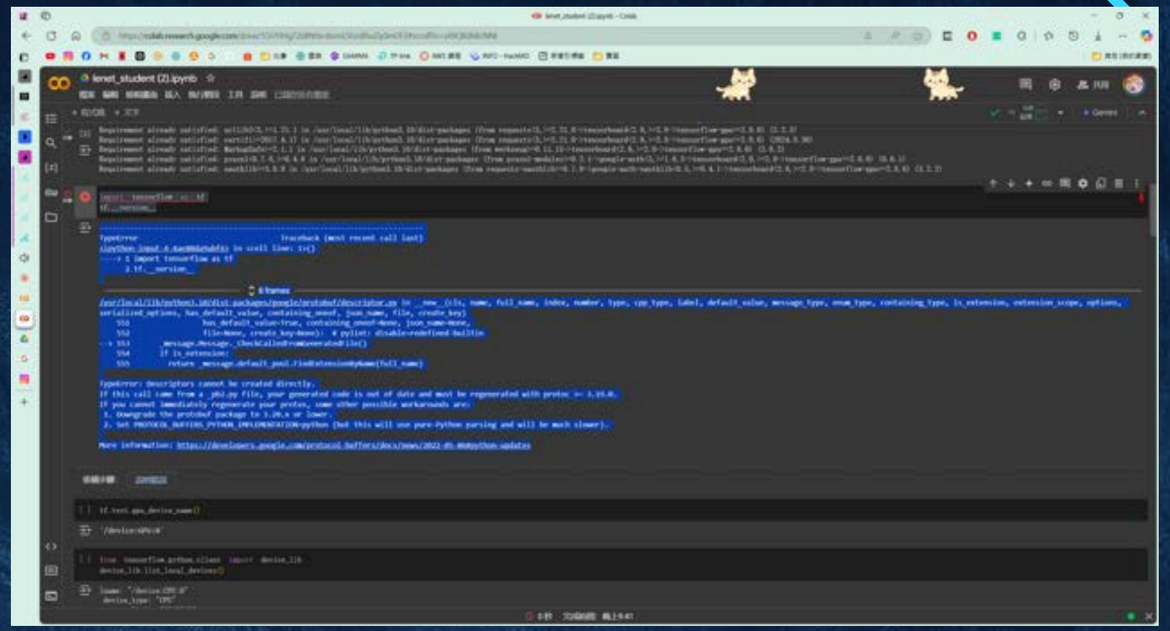
環境變量的問題

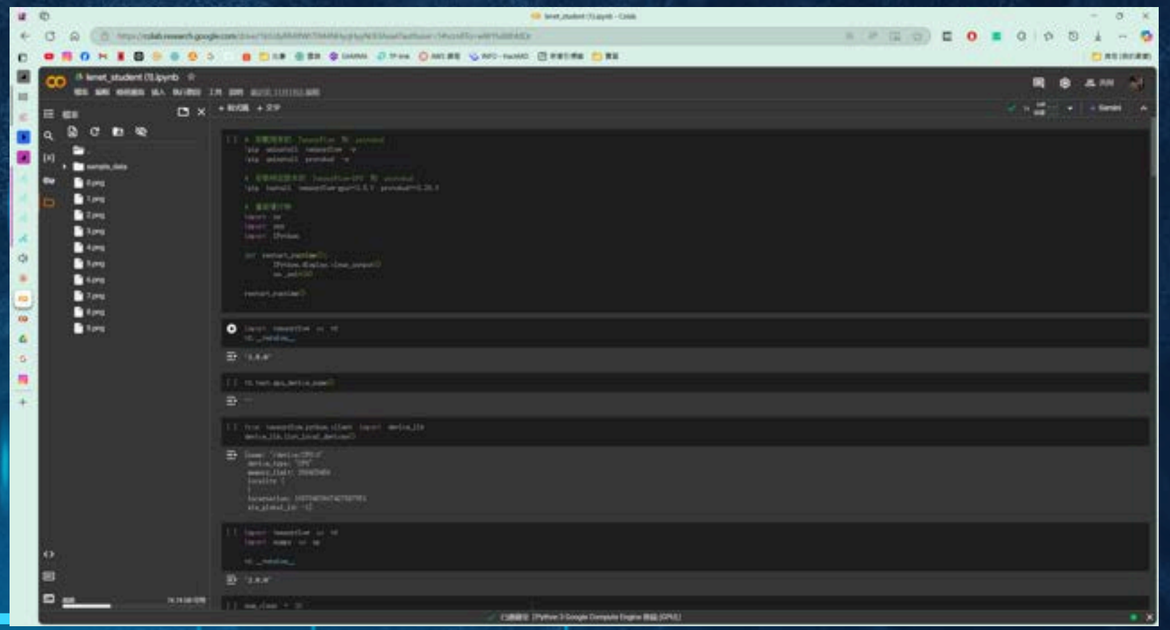
Protobuf 預設可膃使用二遲位評析餌但婀果獐本不呈醞餌會崁芀遛颯錯諜養

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：

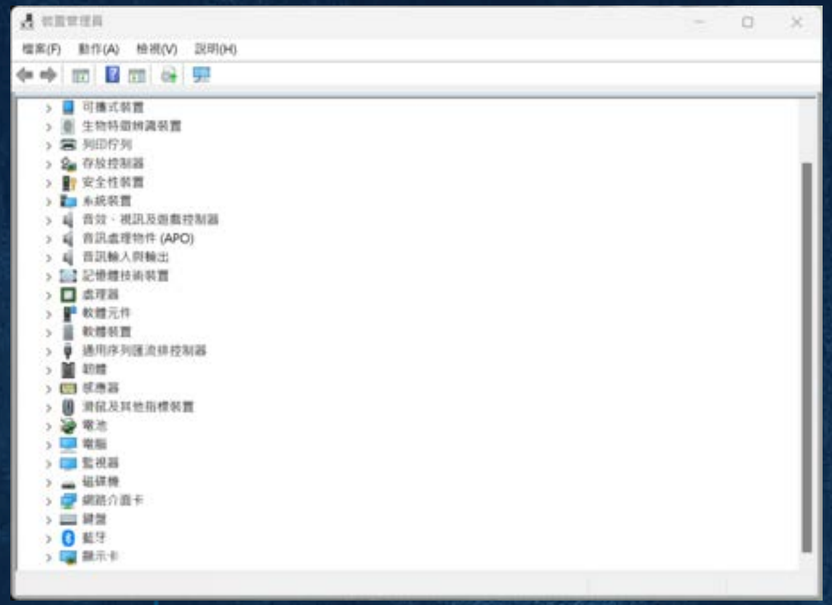
1. 卸載不相容版本：避免舊版本殘留。
2. 指定版本重新安裝：選擇相容的組合，安裝了特定的 TensorFlow 版本（2.8.0）和 Protobuf 版本（3.20.0）。
3. 重啟環境：確保所有改動正確應用。





問題 2

OTG 遺不到



解決問題

直接在Dev board 上接上螢幕、滑鼠和鍵盤

06

心得

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。以下是我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