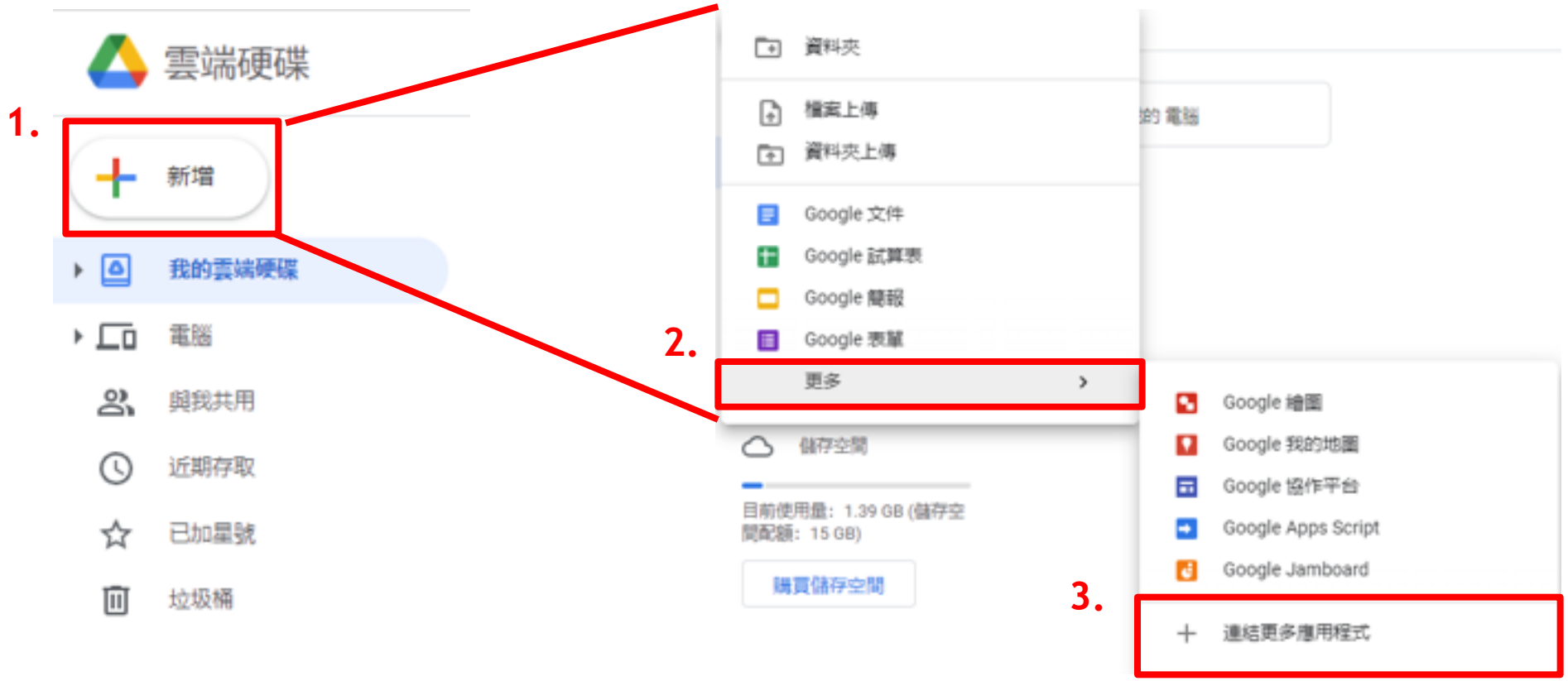
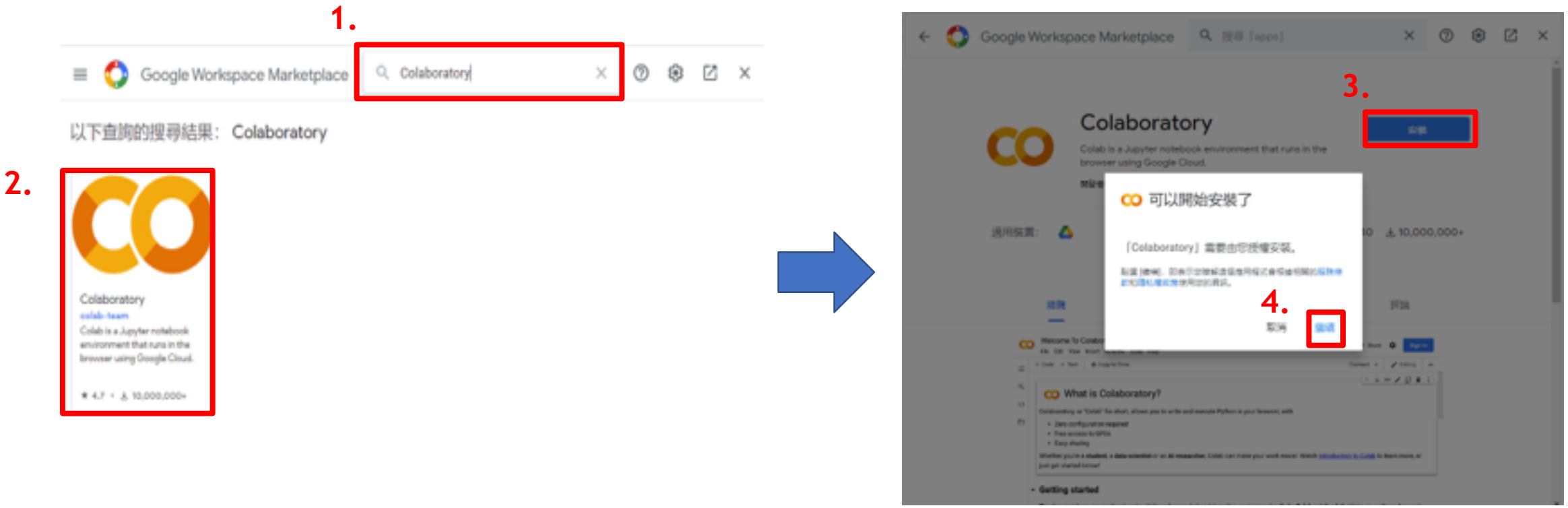
LAB1

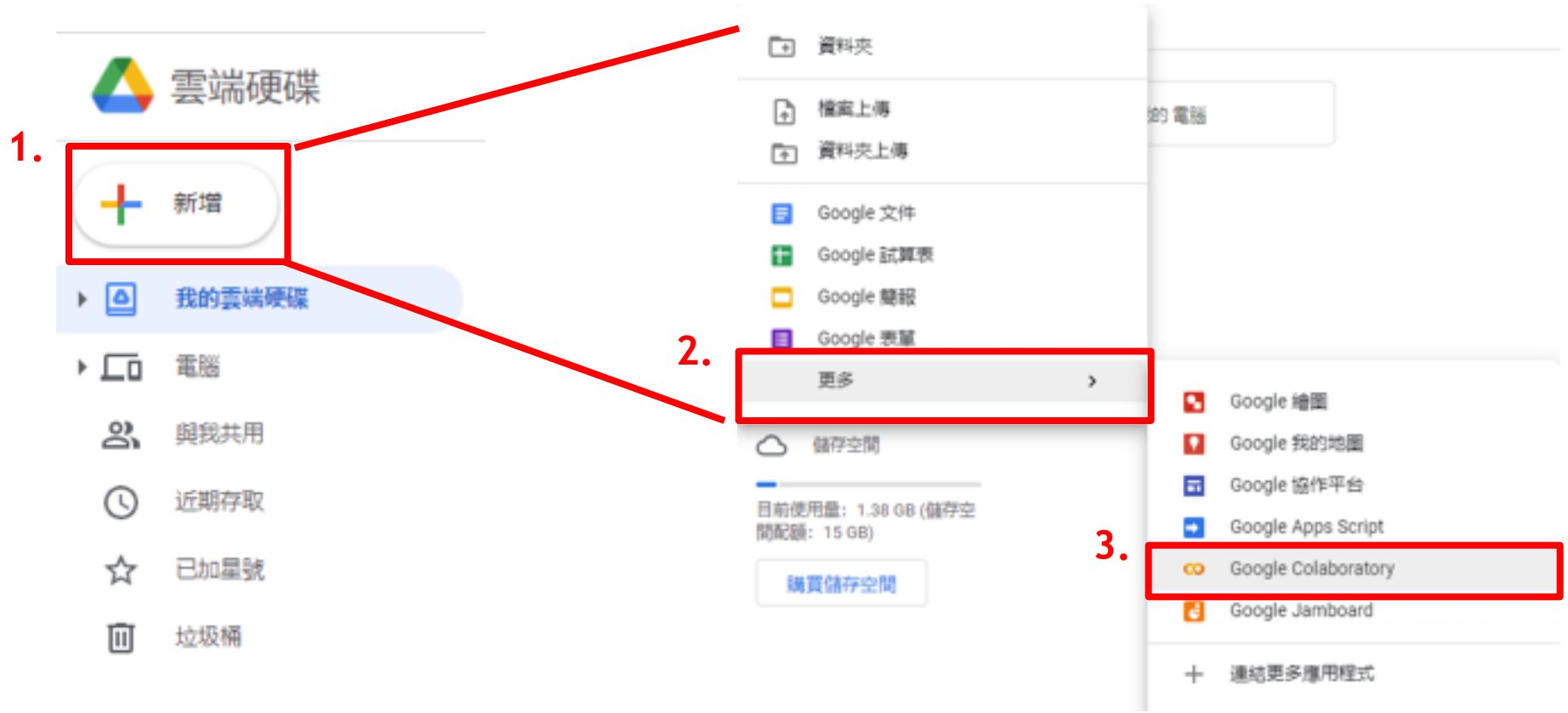
國立雲林科技大學
王斯弘 助理教授 / 前瞻學位學士學程
2021/11/25

Google Colab IDE 使用介紹


1.
雲端硬碟
新增
我的雲端硬碟
電腦
與我共用
近期存取
已加星號
垃圾桶
2.
資料夾
檔案上傳
資料夾上傳
Google 文件
Google 試算表
Google 簡報
Google 表單
更多
儲存空間
目前使用量：1.39 GB (儲存空間配額：15 GB)
購買儲存空間
3.
Google 繪圖
Google 我的地圖
Google 協作平台
Google Apps Script
Google Jamboard
連結更多應用程式

AI計算晶片設計和應用人才培育




雲端硬碟
1.
新增
我的雲端硬碟
電腦
與我共用
近期存取
已加星號
垃圾桶
資料夾
檔案上傳
資料夾上傳
Google 文件
Google 試算表
Google 簡報
Google 表單
2.
更多
儲存空間
目前使用量：1.38 GB (儲存空間配額：15 GB)
購買儲存空間
Google 繪圖
Google 我的地圖
Google 協作平台
Google Apps Script
3.
Google Colaboratory
Google Jamboard
連結更多應用程式

測試 Google Colab IDE

- 輸入 print(“Hellow”)
- 按下程式碼左邊的執行鍵

新增執行格

使用GPU硬體加速器


檔案 編輯 檢視畫面 插入 執行階段 工具 說明 已儲存所有變更
程式碼 + 文字
1.
全部執行 Ctrl+F9
執行上方的儲存格 Ctrl+F8
執行聚焦的儲存格 Ctrl+Enter
執行選取範圍 Ctrl+Shift+Enter
執行下方的儲存格 Ctrl+F10
中斷執行 Ctrl+M I
重新啟動執行階段 Ctrl+M .
重新啟動並執行所有儲存格
恢復原廠設定的執行階段
2.
變更執行階段類型
管理工作階段
查看執行階段記錄
筆記本設定
硬體加速器
None
3.
None
GPU
TPU
固筆記本時，忽略程式碼儲存格輸出內容
4.
取消
儲存

使用GPU硬體加速器

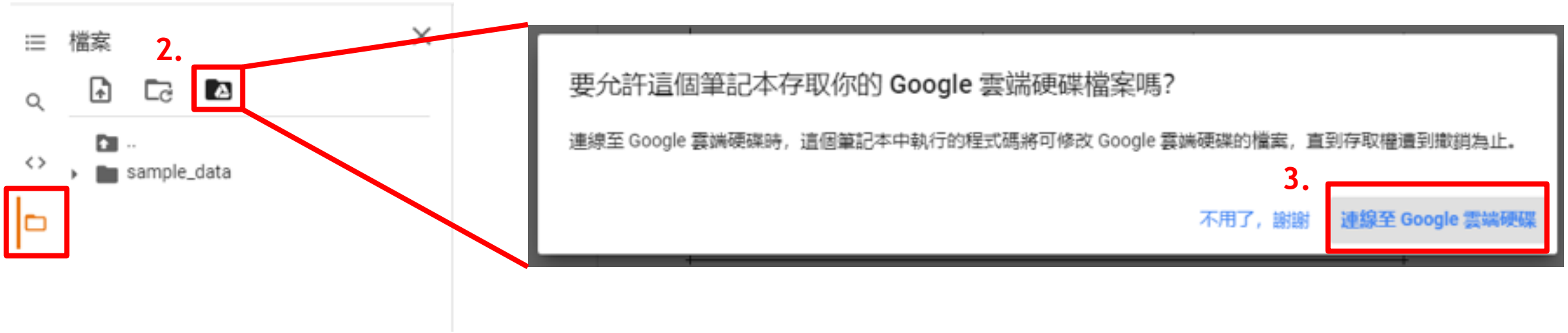
- 輸入 !nvidia-smi 來查看 GPU 目前的狀態

```
[1]  !nvidia-smi

Thu Jul  8 13:09:42 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.42.01    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   35C    P8    28W / 149W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

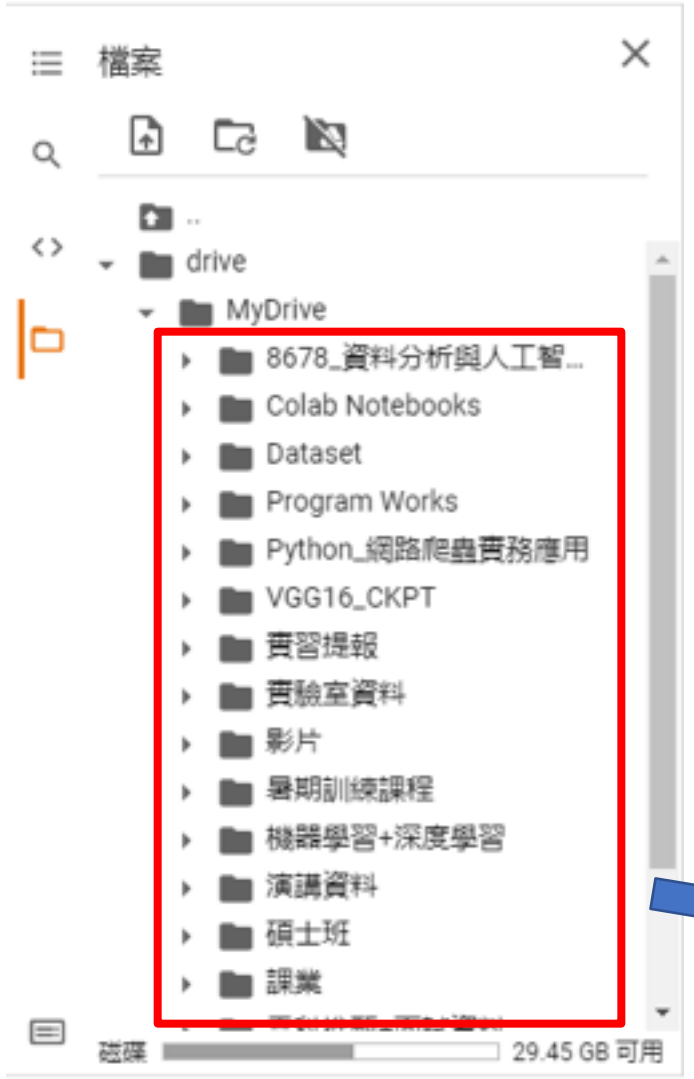
GPU Memory

掛接雲端硬碟


檔案
2.
..
sample_data
1.
要允許這個筆記本存取你的 Google 雲端硬碟檔案嗎?
連線至 Google 雲端硬碟時，這個筆記本中執行的程式碼將可修改 Google 雲端硬碟的檔案，直到存取權遭到撤銷為止。
3.
不用了，謝謝
連線至 Google 雲端硬碟

掛接雲端硬碟

掛接成功後會出現 drive 這個資料夾



你的雲端硬碟資料夾

LeNet MNIST

Import library

import tensorflow as tf

TensorFlow是一個開源軟體庫，用於各種感知和語言理解任務的機器學習。

https://www.tensorflow.org/api_docs/python/tf

Import library

import numpy as np

Numpy 是 Python 的一個重要模組，主要用 於資料處理上。

Import library

import tensorflow.keras as keras

Keras是一個開放原始碼，基於Python高階深度學習的程式庫。他已經將訓練模型的輸入曾、隱藏曾、輸出曾，做好架構。

Import library

import matplotlib.pyplot as plt

Matplotlib是利用Python所實作的繪圖套件， 其中包含兩個最重要的模組-pylab和pyplot。

dataset

mnist = tf.keras.datasets.mnist

#將MNSIT手寫數字資料讀進來

dataset

tf.keras.datasets 是提供 tf.keras.datasets空間的公開Api，就是關於機器學習的數據集，可以直接使用該API獲取並使用數據，有以下幾個數據集：

boston_housing：波斯頓房屋價格回歸數據集
cifar10：CIFAR10小圖像分類數據集
cifar100：CIFAR100小圖像分類數據集
fashion_mnist：Fashion-MNIST 數據集.
imdb：IMDB 分類數據集
mnist：MNIST手寫數字數據集
reuters：路透社主題分類數據集

dataset

```
23 (x_train, y_train), (x_test, y_test) = mnist.load_data()
24 x_train, x_test = x_train / 255.0, x_test / 255.0
25 print('Loading finished')
```

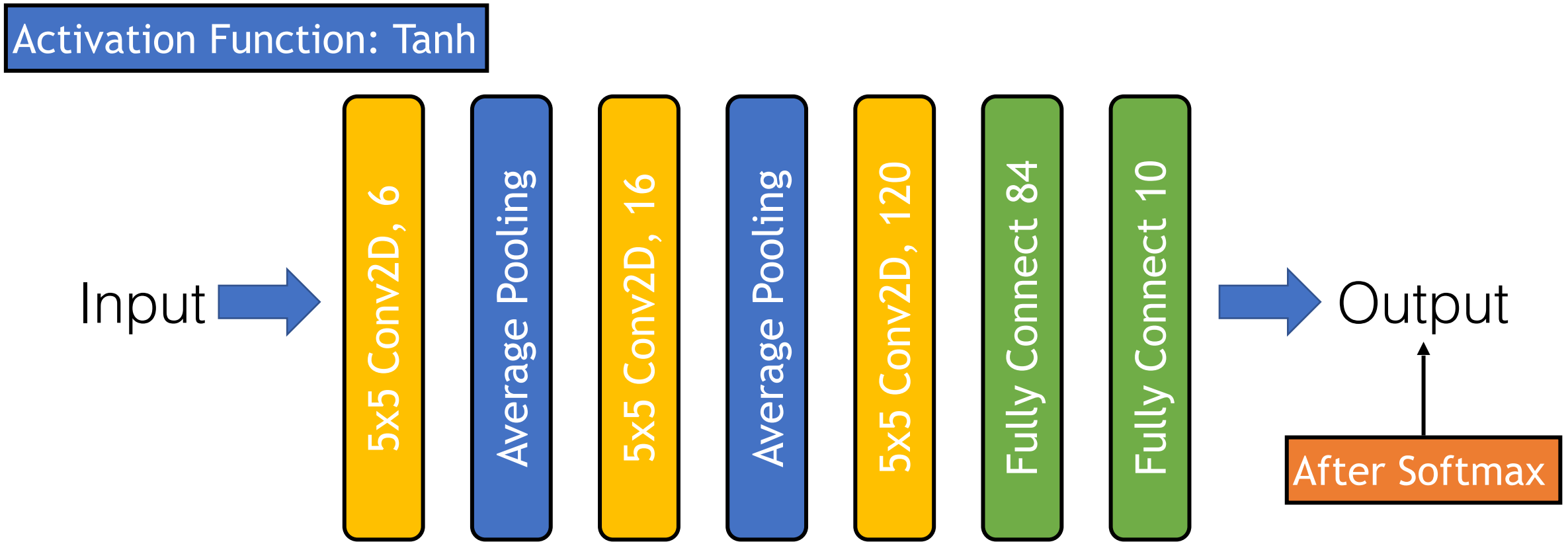
#讀取MNIST手寫資料，並分成train和test。
#在訓練前做正規化(Normalization)，將影像數值縮至0~1之間。

訓練

```
model = Sequential()
model.add(Conv2D(filters=6, kernel_size=(5,5), strides=(1,1), activation='tanh', input_shape=(28,28,1)))
model.add(AveragePooling2D(pool_size=(2,2), strides=(2,2)))
model.add(Conv2D(filters=16, kernel_size=(5,5),strides=(1,1), activation='tanh'))
model.add(AveragePooling2D(pool_size=(2,2), strides=(2,2)))
model.add(Conv2D(filters=10, kernel_size=(3,3), strides=(1,1), padding='same',activation='tanh'))
model.add(Flatten())
model.add(Dense(units=120, activation='tanh'))
model.add(Dense(units=84, activation='tanh'))
model.add(Dense(units=10, activation='softmax'))
```

LeNet網路架構

Lab-Network Structure



LeNet-5網路架構請參考以下的網址:
https://towardsdatascience.com/understanding-and-implementing-lenet-5-cnn-architecture-deep-learning-a2d531ebc342

訓練

```
4 # 訓練網路模型
5 history = model.fit(x_train, y_train,
6           batch_size=32,
7           epochs=10,
8           verbose=1,
9           validation_data=(x_test, y_test))
10 # 儲存h5權重檔
11 model.save('MNIST_ep10.h5')
```

model.fit(訓練集的輸入特徵，
訓練集的標籤，
batch_size, #每一個batch的大小
epochs, #迭代次數
validation_data = (測試集的輸入特徵，測試集的標籤），
validation_split = 從測試集中劃分多少比例給訓練集，
validation_freq = 測試的epoch間隔數

訓練

```
Epoch 1/10
1875/1875 [==============================] – 40s 5ms/step – loss: 0.2480 – accuracy: 0.9262 – val_loss: 0.1221 – val_accuracy: 0.9618
Epoch 2/10
1875/1875 [==============================] – 9s 5ms/step – loss: 0.1001 – accuracy: 0.9695 – val_loss: 0.0934 – val_accuracy: 0.9709
Epoch 3/10
1875/1875 [==============================] – 10s 5ms/step – loss: 0.0696 – accuracy: 0.9788 – val_loss: 0.0616 – val_accuracy: 0.9801
Epoch 4/10
1875/1875 [==============================] – 10s 5ms/step – loss: 0.0535 – accuracy: 0.9832 – val_loss: 0.0634 – val_accuracy: 0.9802
Epoch 5/10
1875/1875 [==============================] – 10s 5ms/step – loss: 0.0421 – accuracy: 0.9869 – val_loss: 0.0554 – val_accuracy: 0.9816
Epoch 6/10
1875/1875 [==============================] – 9s 5ms/step – loss: 0.0364 – accuracy: 0.9879 – val_loss: 0.0581 – val_accuracy: 0.9821
Epoch 7/10
1875/1875 [==============================] – 9s 5ms/step – loss: 0.0311 – accuracy: 0.9896 – val_loss: 0.0567 – val_accuracy: 0.9830
Epoch 8/10
1875/1875 [==============================] – 10s 5ms/step – loss: 0.0275 – accuracy: 0.9908 – val_loss: 0.0519 – val_accuracy: 0.9838
Epoch 9/10
1875/1875 [==============================] – 9s 5ms/step – loss: 0.0213 – accuracy: 0.9933 – val_loss: 0.0445 – val_accuracy: 0.9871
Epoch 10/10
1875/1875 [==============================] – 9s 5ms/step – loss: 0.0196 – accuracy: 0.9934 – val_loss: 0.0493 – val_accuracy: 0.9859
```

訓練

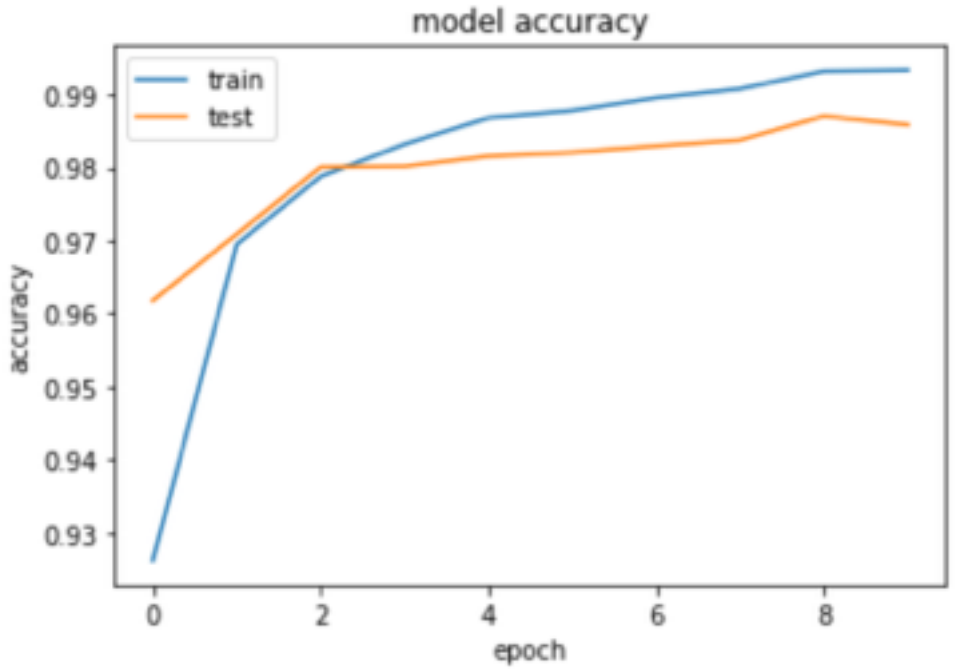
model.summary()
查看參數

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 24, 24, 6)	156
average_pooling2d (AverageP ooling2D)	(None, 12, 12, 6)	0
conv2d_1 (Conv2D)	(None, 8, 8, 16)	2416
average_pooling2d_1 (Averag ePooling2D)	(None, 4, 4, 16)	0
conv2d_2 (Conv2D)	(None, 4, 4, 10)	1450
flatten (Flatten)	(None, 160)	0
dense (Dense)	(None, 120)	19320
dense_1 (Dense)	(None, 84)	10164
dense_2 (Dense)	(None, 10)	850

Total params: 34,356
Trainable params: 34,356
Non-trainable params: 0

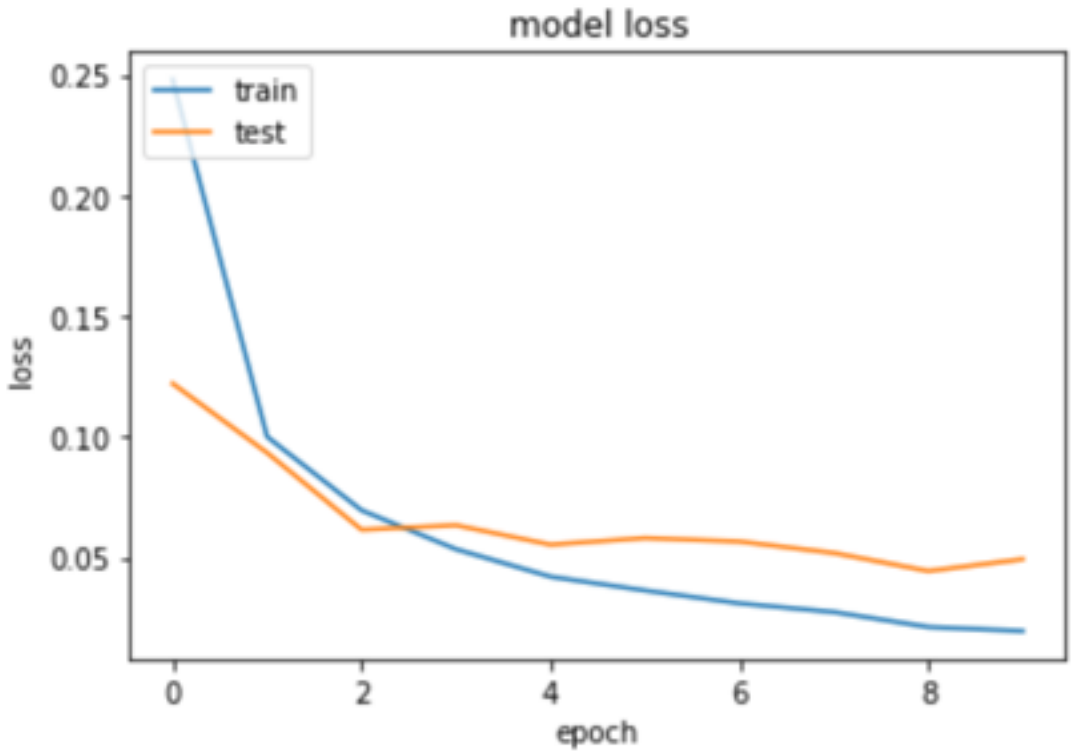
結果

```
13 # summarize history for accuracy
14 plt.plot(history.history['accuracy'])
15 plt.plot(history.history['val_accuracy'])
16 plt.title('model accuracy')
17 plt.ylabel('accuracy')
18 plt.xlabel('epoch')
19 plt.legend(['train', 'test'], loc='upper left')
20 plt.savefig('[Accuracy]20200805_MNIST_ep10_b128.jpg')
21 plt.show()
```



結果

```
24 # summarize history for loss plt.plot(history.history['loss']) plt.plot(history.history['val_loss']) plt.title('model loss')
25 plt.plot(history.history['loss'])
26 plt.plot(history.history['val_loss'])
27
28 plt.title('model loss')
29 plt.ylabel('loss')
30 plt.xlabel('epoch')
31
32 plt.legend(['train', 'test'], loc='upper left')
33 plt.savefig('[Loss]20200805_MNIST_ep10_b128.jpg')
34 plt.show()
```



END