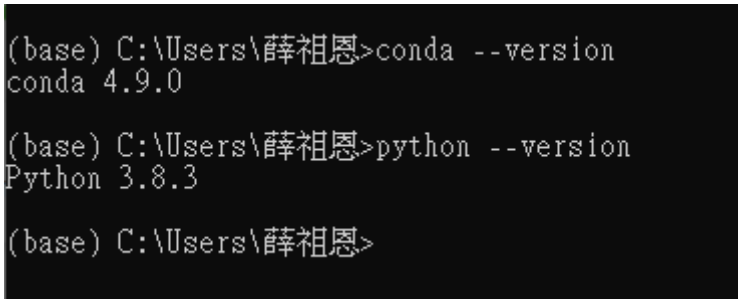
機器學習

開發環境記錄過程進度報告

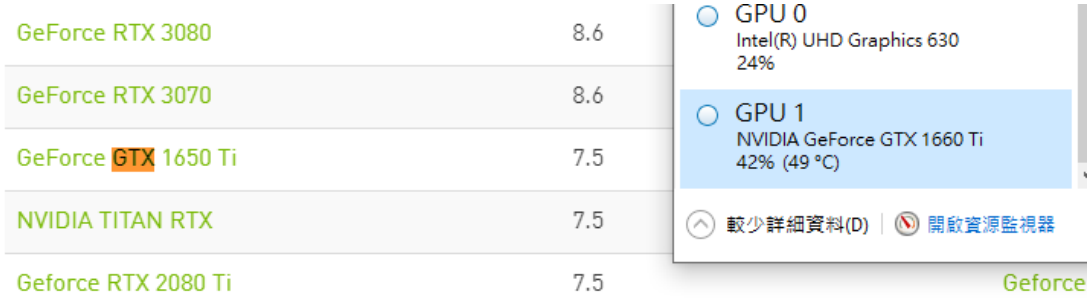
學生學號：410821305

學生姓名：薛祖恩

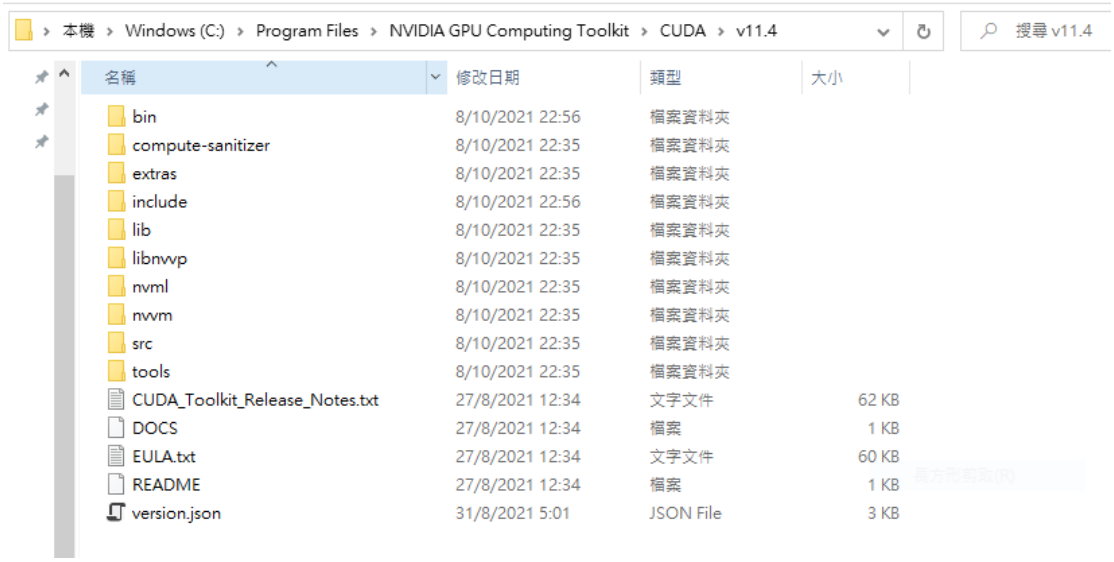
一、 (1)確認 anaconda 以及 python 版本已經安裝完畢



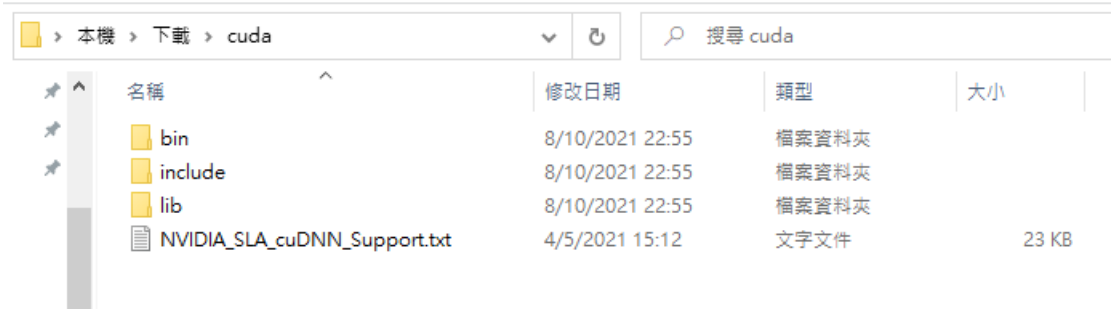
二、 (1)確認 GPU Compute Capability



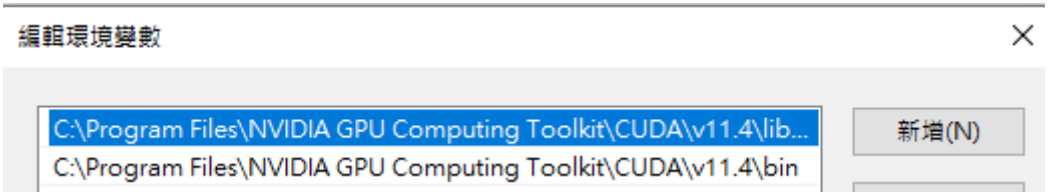
三、 (1)確認 CUDA 已經安裝至本地



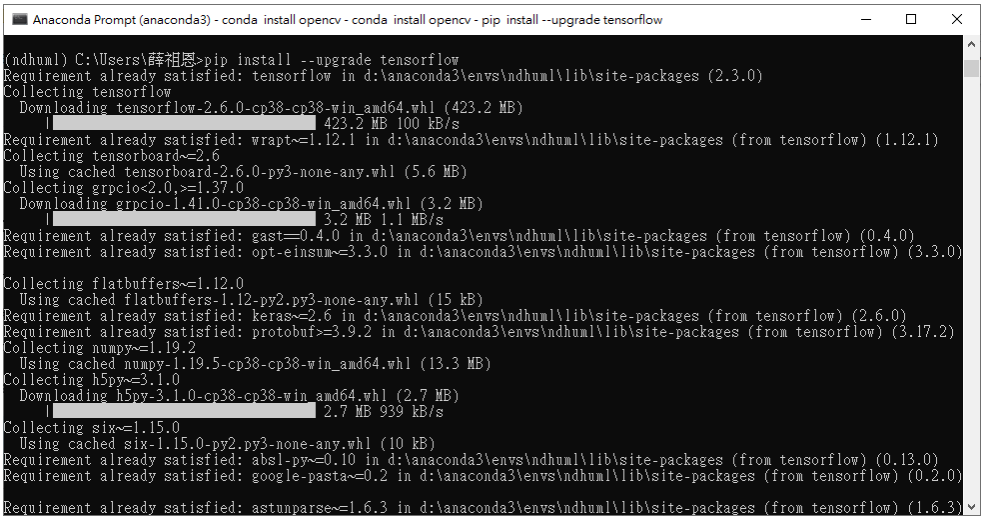
四、 (1)安裝相對應 cuDNN 並且取代 CUDA 中的 bin, include 以及 lib



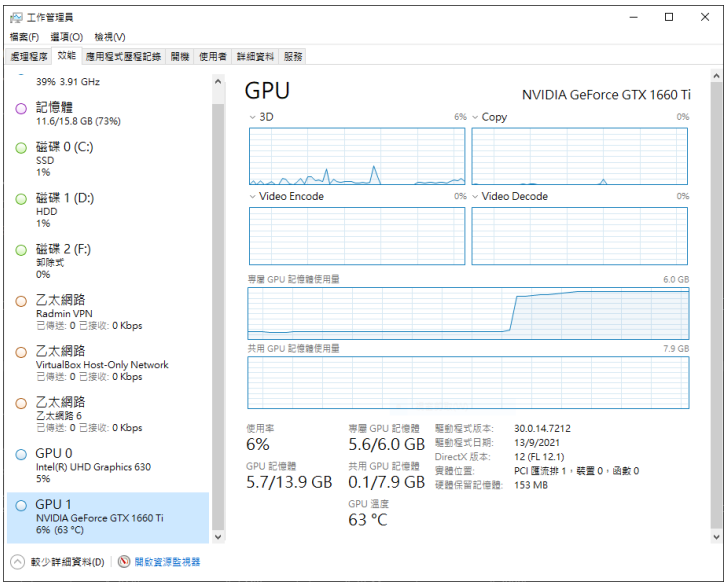
五、 (1)新增環境變數



六、 (1)更新 tensorflow



七、 (1)執行 PPT 上面的程式碼，檢查 GPU 運作狀況



八、 (2)在 Colab 執行範例程式

```
import cv2
import numpy as np

from keras.applications.vgg16 import VGG16
from keras.layers import Input
from keras.layers import Flatten
from keras.layers import Dense
from keras.layers import Dropout
from keras.models import Model
from keras.utils import np_utils
from keras.datasets import mnist

epochs = 10
batch_size = 50
row_col = 48

# 原始的 MNIST 是 6000 筆 28*28 灰階
def load_data():
    (X_train, y_train), (X_test, y_test) = mnist.load_data()
    X_train, y_train = X_train[:5000], y_train[:5000]
    X_test, y_test = X_test[5000:6000], y_test[5000:6000]
    X_train = [cv2.cvtColor(cv2.resize(i, (row_col, row_col)), cv2.COLOR_GRAY2RGB)
                    for i in X_train]
    X_train = np.concatenate([arr[np.newaxis] for arr in X_train]).astype('float32')
    X_test = [cv2.cvtColor(cv2.resize(i, (row_col, row_col)), cv2.COLOR_GRAY2RGB)
                    for i in X_test]
    X_test = np.concatenate([arr[np.newaxis] for arr in X_test]).astype('float32')

    X_train = X_train / 255
    X_test = X_test / 255

    y_train_ohe = np_utils.to_categorical(y_train, 10)
    y_test_ohe = np_utils.to_categorical(y_test, 10)

    return (X_train, y_train_ohe), (X_test, y_test_ohe)
```

```
    return (X_train, y_train_ohe), (X_test, y_test_ohe)


def load_model():

    base_network = VGG16(include_top=False, weights='imagenet', input_shape=(row_col, row_col, 3))

    # 凍結預設的參數
    for layer in base_network.layers:
        layer.trainable = False

    # 接上自行定義的全連結層
    model = Flatten()(base_network.output)
    model = Dense(4096, activation='relu', name='full_connect_1')(model)
    model = Dense(4096, activation='relu', name='full_connect_2')(model)
    model = Dropout(0.5)(model)
    model = Dense(10, activation='softmax', name='prediction')(model)
    model = Model(base_network.input, model, name='my_model')

    return model


# 載入模型
model = load_model()
model.summary()

# 載入訓練資料
(x_train, y_train_ohe), (x_test, y_test_ohe) = load_data()
print('Train Size:', x_train.shape)
print('Test Size:', x_test.shape)

#定義 loss function
from tensorflow.keras.optimizers import SGD
sgd = SGD(lr=0.05, decay=1e-5)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

# 開始訓練
model.fit(x_train, y_train_ohe, validation_data=(x_test, y_test_ohe), epochs=epochs, batch_size=batch_size)

# 輸出結果
print('Train Acc:', model.evaluate(x_train, y_train_ohe)[1])
print('Test Acc:', model.evaluate(x_test, y_test_ohe)[1])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 1s 0us/step
58900480/58889256 [==============================] - 1s 0us/step
Model: "my_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 48, 48, 3)]       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 48, 48, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 48, 48, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 24, 24, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 24, 24, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 24, 24, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 12, 12, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 12, 12, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 12, 12, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 12, 12, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 6, 6, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 6, 6, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 6, 6, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 6, 6, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 3, 3, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
full_connect_1 (Dense)       (None, 4096)              2101248
_________________________________________________________________
full_connect_2 (Dense)       (None, 4096)              16781312
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
prediction (Dense)           (None, 10)                40970
=================================================================
Total params: 33,638,218
Trainable params: 18,923,530
Non-trainable params: 14,714,688
_________________________________________________________________
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
Train Size: (5000, 48, 48, 3)
Test Size: (1000, 48, 48, 3)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/optimizer_v2.py:356: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  "The `lr` argument is deprecated, use `learning_rate` instead.")
Epoch 1/10
100/100 [==============================] - 35s 58ms/step - loss: 1.3578 - accuracy: 0.5510 - val_loss: 0.7878 - val_accuracy: 0.7480
Epoch 2/10
100/100 [==============================] - 6s 55ms/step - loss: 0.7020 - accuracy: 0.7820 - val_loss: 0.5256 - val_accuracy: 0.7940
Epoch 3/10
100/100 [==============================] - 5s 50ms/step - loss: 0.5693 - accuracy: 0.8066 - val_loss: 0.5314 - val_accuracy: 0.7890
Epoch 4/10
100/100 [==============================] - 6s 55ms/step - loss: 0.4748 - accuracy: 0.8442 - val_loss: 0.4164 - val_accuracy: 0.8530
Epoch 5/10
100/100 [==============================] - 5s 55ms/step - loss: 0.4237 - accuracy: 0.8562 - val_loss: 0.3484 - val_accuracy: 0.8790
Epoch 6/10
100/100 [==============================] - 5s 49ms/step - loss: 0.3728 - accuracy: 0.8810 - val_loss: 0.5765 - val_accuracy: 0.7650
Epoch 7/10
100/100 [==============================] - 5s 49ms/step - loss: 0.3584 - accuracy: 0.8792 - val_loss: 0.3478 - val_accuracy: 0.8750
Epoch 8/10
100/100 [==============================] - 5s 49ms/step - loss: 0.3182 - accuracy: 0.8972 - val_loss: 0.2433 - val_accuracy: 0.9270
Epoch 9/10
100/100 [==============================] - 6s 55ms/step - loss: 0.2928 - accuracy: 0.9038 - val_loss: 0.4005 - val_accuracy: 0.8710
Epoch 10/10
100/100 [==============================] - 6s 55ms/step - loss: 0.2765 - accuracy: 0.9108 - val_loss: 0.3097 - val_accuracy: 0.8790
157/157 [==============================] - 6s 36ms/step - loss: 0.2872 - accuracy: 0.8970
Train Acc: 0.8970000147819519
32/32 [==============================] - 1s 29ms/step - loss: 0.3097 - accuracy: 0.8790
Test Acc: 0.8790000081062317
```