1. 前言

我上學期有修過電腦視覺與深度學習，其中有作業是用pytorch實作一個模型，因此對電腦視覺相關深度學習 (尤其是 pytorch)已經有一定程度理解。

我常用電腦 (實驗室電腦) 無法使用cuda，而colab有時間限制不能讓我隨便測試，我自己的筆電雖然有cuda，但是平常待在實驗室不常有時間使用，因此整個作業要在實驗室電腦、colab、自己筆電三個不同環境上跑。

1. 初始版本

因為連會不會中途噴錯誤都不知道，第一個板本主要目標是能做出來就好。

1. 資料前處理

因為第一次用keras，連能不能成功跑起來都不知道，因此只做最小程度的前處理。

前處理包含進行one hot和改變資料集維度，這邊與pytorch不同的是這邊label形狀是(X,1,1,43) 而非(X,43)，因此要再多一步。

Y\_OneHot = utils.to\_categorical(Y, num\_classes=num\_classes);

Y\_Reshape = Y\_OneHot.reshape((-1, 1, 1, num\_classes));

1. 建立模型

使用最基本的Lenet-5

def Lenet(n\_classes: int) -> Model:

    model = keras.Sequential();

    model.add(BatchNormalization());

    model.add(Conv2D(6, (5, 5), activation=tanh));

    model.add(MaxPooling2D((2, 2)));

    model.add(Conv2D(16, (5, 5), activation=tanh));

    model.add(MaxPooling2D((2, 2)));

    model.add(Conv2D(120, (5, 5), activation=tanh));

    model.add(Dense(84, activation=tanh));

    model.add(Dense(n\_classes, activation=softmax));

    return model;

這個模型幾乎與原先路文上的一樣，只有改變輸出節點數、輸出層改為全連接(而非高斯連接)、在卷積層(而非池化層)使用非線性。

因為是分類任務，模型loss採用cross entropy。

模型optimizer採用sgd。根據上學期經驗，sgd很慢、準度很低，但是如果模型或其他地方搞砸 (例如沒初始化權重) 的話sgd至少還能跑出結果 (準度大於平均)，adam 剛跑就會直接跑出NaN。

1. 訓練模型

使用的超參數為:

epoch=10, batch\_size=16。

和pytorch (或tensorflow) 很不一樣的是訓練過程不是迴圈，而是把超參數餵進model.fit之後直接跑完全程。

1. 測試結果

我知道測試集不該隨時拿出來看，但是因為訓練過程中驗證集的loss看起來還不錯 (大約0.17)，我還是拿出來看了一下。

結果:

Valid loss = 0.2038 (沒有accuracy，我不知道怎麼叫出來)。

Test loss: 0.2243 (同樣沒有accuracy)。

Inference: 那10張全部判斷成第三個，機率都是0.494。

看到這裡我感覺前面這些訓練都是打假球，怎麼會訓練時loss都有降低，結果那10張不只全錯，得到的數值還一模一樣。

1. 改良版本

這個版本主要改善可移植性 (畢竟我要在三個不同環境上跑)。

1. 環境設定

我知道這很枝微末節，浪費的時間大概也比省的時間多，但是為了未來的我方便，我還是做個簡單版本控制系統，在每次全部執行的時候一起執行，在最新版本的時候提早結束，並跳過已安裝完成的部分。

這裡不講細節，但是大概念是先看env.txt內的版本內容，如果不是最新就跑安裝套件，並且檢查必要資料夾或檔案是否都存在。如果是Colab的話要再掛載雲端硬碟和複製到本地。

(下面是檢查安裝套件的函式，對於不能輸入的vscode來說很有用)

def install\_package(package\_name: str) -> None:

    if not importlib.util.find\_spec(package\_name):

        command = ["pip", "install", package\_name];

        proc = subprocess.Popen(command, stdin=subprocess.PIPE, stdout=subprocess.PIPE, stderr=subprocess.STDOUT);

        \_ = proc.communicate(input='y'.encode())[0];

1. 建立模型

因為lenet5模型架構簡單，對於模型分類有一定極限，因此這邊再新增幾個模型。