Lab2報告

組員：108552005 高儀津(Luca\_Kao) , 108552017 申自強(Arthur.Shen)

**Work Flow**

1. 前言

由於本次資料集有58.82 GB 過大光下載在組員間就花了兩天的時間 , 期間一邊下載一邊思考這次的作業有別於上次單純的csv 檔, 還外加圖片檔, 所以資料集才會這麼大 , 在研究資料集圖片後我們這組決定同組的兩個人使用不同的作法 , 因為兩個人擁有的硬體資源不同 :

108552005 高儀津 : 使用本地訓練(Nvdia GPU 1070)和跟實驗室借用 Nvdia GPU 2070 , 採用Pytorch的方式做運算 .

108552017 曾自強 : 主要使用kaggle的資源和本地訓練居多 , 採用TensorFlow的方式做運算 .

1. 資料預處理

因為本次的Train的植物分類圖片集內有百萬張圖片 , 因此我們有特別研究Json檔內的特徵有32093種 , 而且裡面圖片依照科(famliy) , 屬(genus) , 種(name)的方面每層資料擺放 , 並有部分種類的圖片是很少甚至只有一張發現有imbalance的現象 , 所以前處理把多張特徵表格依照category\_id方式相連接處理後:

用 sklearn的LabelEncoder處理 , 這種編碼方式的目的是為了將類別 (categorical)或是文字(text)的資料轉換成數字，而讓程式能夠更好的去理解及運算。

Label encoding : 把每個類別 mapping 到某個整數，不會增加新欄位 .

**3.**  模型訓練和預測

PyTorch - 參考108552005\_PyTorch\_Train.ipynb , 108552005\_PyTorch\_Predit.ipynb

3.1 自定義TrainDataset和transforms

- 解析度 HEIGHT , WEIGHT = 128 PIXEL

* 透過TrainDataset將資料集處理後用PyTorch的ToTensorV2轉成張量 , 然後載入資料放DataLoader

3.2 DataLoader

batch\_size = 256 批次訓練的大小

train\_loader = DataLoader(train\_dataset, num\_workers=4, batch\_size=batch\_size, shuffle=True)

num\_workers 使用多線程的方式加速工作時間

3.2 train valid split

StratifiedKFold 分層取樣交叉切分，確保訓練集，測試集中各類別樣本的比例與原始資料集中相同。

3.3 選用優化器優化(Optimizer)

- Adam

- SGD

- CrossEntropyLoss

3.4 建立CNN model

- resnet18

- epoch = 20 次

4. 討論

本次實作的過程中 , 發現此專案需要不錯的設備長時間訓練n個epoch來做觀察 , 因此會花費很多時間在觀察跑epoch上 , 而 model 選用上 ,我們有測試過 Alexnet , resnet50 , VGG16 來做測試 , 但效果沒有resnet18 好 , 因此後來主要還是使用resnet18 做主要的model , 且當中其實遇到很多次記憶體不足調整了batch\_size 和 解析度[out of memory...] , 最後是借用了實驗室的GPU跑20次以上來預測到最後的分數0.31585 , Ranking = 30 , 有發現在不處理植物分類圖集的資料imbalance的狀況下 , 最後花費大量時間跑的epoch , 成績都不是很理想[epoch的後期的分數並無明顯的改善] , 猜測可能是imbalance的狀況造成 , 因此以後如果還要處理類似問題的話 , 可能要思考如何處理此問題 , 像是重新組合不均衡數據 , 使資料集均衡又或者複製和合成少數部分的樣本, 使之和多數部分差不多數量. 也可以砍掉一些多數部分, 使兩者數量差不多.

參考:

1. <https://morvanzhou.github.io/tutorials/machine-learning/torch/3-06-optimizer/>
2. <https://www.kaggle.com/yasufuminakama/herbarium-2020-pytorch-resnet18-train>
3. <https://www.kaggle.com/yasufuminakama/herbarium-2020-pytorch-resnet18-inference>