**繁體中文場景文字辨識競賽－**

**進階：繁體中文場景文字辨識**

**報告文件**

[環境](#環境) ………………………………………………………………… P.2

[演算方法與模型架構](#演算方法) ……………………………………………… P.2

[資料處理](#資料處理) …………………………………………………………… P.3

[訓練方式](#訓練方式) …………………………………………………………… P.4

[分析與結論](#分析與結論) ………………………………………………………… P.5

[程式碼](#程式碼) ……………………………………………………………… P.8

[使用的外部資源與參考文獻](#使用的外部資源與參考文獻) ……………………………………… P.8

1. 環境
2. 作業系統

* Window 10
* Ubuntu 18.04.5 LTS (Google Colaboratory)

1. 程式語言

* Python 3.8.8 (Anaconda 2021.05)
* Python 3.7.12(Google Colaboratory)

1. 套件(函式庫)

* Tensorflow 2.2.0
* Tensorflow-gpu 2.2.0
* Keras 2.4.3
* Imageai 2.1.6
* Opencv-python 4.5.3.56
* Numpy 1.19.3

1. 預訓練模型

* InceptionResNetV2, 使用Keras內建的pre-training on ImageNet

1. 額外資料集

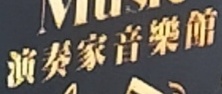
* 自行蒐集的街景圖製作成偵測樣本以及分類樣本

1. 演算方法與模型架構

我們組按此次題目要求大致將整個模型架構分成兩層，第一層是偵測模型，我們選用ImageAI內的YoloV3模型，主要目的在於對應到題目中找出繁體字所在座標，次要目的則在減少下一個模型輸入的噪點，盡量避免丟進其他圖示、英文數字等不符合題目但又可能被當作字的物件；第二層則是分類模型，選用Keras內建Model API的InceptionResNetV2，要做到的功能就是將前一層模型得出的繁體字進行分類，將圖片轉文字以繳交。

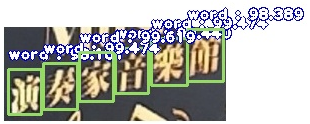
其中ImageAI的模型參數皆維持預設，改動僅調整訓練參數，詳情會在訓練方式中提及。InceptionResNetV2則有較多改動，模型架構部分我們選擇拋棄頂層的fully connected layer以及凍結前兩層網路(一方面給GPU記憶體降壓，另一方面是實驗結果這樣會高出2%以上的準確度)，並在最後加上Flatten Layer + Dropout Layer(0.5) + Dense Layer(1111是分類數量, 激勵函數使用softmax)。

模型效果流程



裁切

偵測模型



C:\Users\brian\Desktop\場景文字\Classification\ok476.jpgC:\Users\brian\Desktop\場景文字\Classification\ok440.jpgC:\Users\brian\Desktop\場景文字\Classification\ok405.jpg

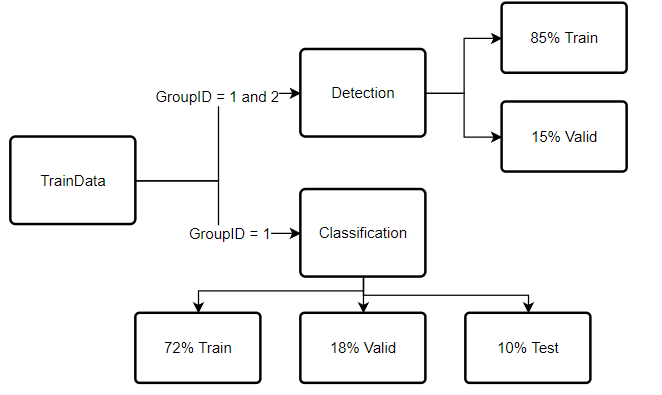
分類模型

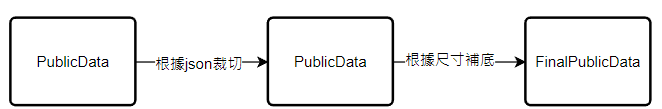
裁切

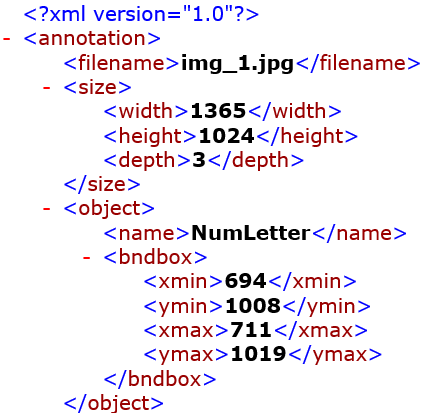
C:\Users\brian\Desktop\場景文字\Classification\ok584.jpgC:\Users\brian\Desktop\場景文字\Classification\ok548.jpgC:\Users\brian\Desktop\場景文字\Classification\ok512.jpg

1. 資料處理

Train & Public Data處理流程圖





1. 偵測訓練

由於我們偵測模型所使用的是ImageAI套件中的Yolov3，因此必須先寫一隻程式將官方給的json變成該套件讀取註解檔的要求(同Labelimg的VOC註解格式)，將其中的物件分類、座標位置給寫進xml檔案，變成如右圖的模樣。

其中座標位置必須構成一個矩形，因此在轉換過程中，我們的程式邏輯是從Json中的四組座標中找出X值最大最小與Y值最大最小來構成矩形，另外由於我們為了訓練的目的，我們挑選的訓練偵測的物件有分類1(中文字元)跟分類2(英數字串)，我們將其分類重新命名為Word跟NumberLetter，在這邊簡單提一下想法是因為我們認為我們的任務目標只是認出繁體中文字元，但有時候會跟數字或英文搞混，因此多放入英數字串的物件去做學習，這樣只要判斷出英數字串就能直接淘汰掉，會比單獨訓練中文字元效果好。

最後進行訓練測試資料的分配，將85%放到Train Data，15%放入Valid Data，到此就做好偵測訓練的前處理。

1. 分類訓練

分類部分我們利用主辦方給的Train Data的json檔案，並使用之前寫的轉換座標成矩形的Function，寫一支裁切程式將我們的中文字元部分全部切出來，並按照原本json檔案給的label標籤(就是該字的中文)去做命名，可以得到原本48591個樣本，但因為有些分類樣本只有一個，因此刪減後得到979個分類共43959個字，依照習慣將其按照比例10%當作Test Data，然後剩下的部分按照80%作為Train Data(佔全部72%)、20%為Valid Data(佔全部18%)，並在後續透過補充蒐集到的新字樣本按照比例分配進去，最後是1111個分類共46185個字，涵蓋public題目中約90%的出現字次數。

1. 偵測物的處理

由於電腦效能的取捨，我們會先將public data中的目標物建按照csv檔案事先裁切下來，並透過補底(方型白底)的方式去凸顯目標達到輔助偵測的效果，詳細效果我們會放在分析與結論去做解釋。

1. 字元辨識的處理

由於分類模型我們是設定input size = 100\*100，因此在偵測模型偵測出字之後，我們會進行縮放(使用cv2.resize, interpolation=cv2.INTER\_CUBIC), 縮放至相對應的大小，這是我們對比其他幾個縮放模式的效果跟準確度得到的結果，大約比預設的縮放方式提高2-3%的準確度。

1. 訓練方式
2. Detectin

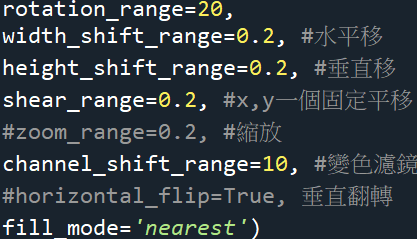
使用ImageAI套件。在欲進行操作的資料夾內，將前面做好處理的資料按照85%、15%的比例分別創建並放入train跟validation兩個資料，接著使用官網的Custom Detection Model Training[[1]](#footnote-1)，參數部分由於顯示卡使用1060，顯卡記憶體只有6G，因此batch size只能設定為4(亦有使用colab跑8，速度節省四分之一，但由於有使用時間限制，因此完成版還是依靠自家主機跑)；num\_experiments設定至少超過30，設定只要不要太小都可是因為每次進步都會存下權重；並且我們沒有特別使用預訓練權重檔。

訓練可隨時中斷，而訓練開始之後會產生多個資料夾，其中最重要的兩個資料夾分別是json跟model，前者會放yolov3的Anchor box設定檔(ImageAI使用k-mean計算)，後者則是存放yolov3的.h5權重檔。

1. Classification

使用Keras套件內建Model API的InceptionResNetV2。只需要將前面處理好的照片按比例分配，我們為了要更好的評估模型，在這邊我們使用前處理做好的10%(Test)、18%(Valid)、72%(Train)，使用train跟valid去訓練，並拿最後test的資料作最後階段的驗收與模型選擇，輸入資料須設定縮放大小到100\*100，並設定batch size為16、最後的dense layer為1111(符合分類數)，Epoch設定為2是因為我們使用迴圈去進行訓練，這樣才能每兩圈存下一個權重檔做Test，以方便內部先選出較好的權重，避免浪費一天三次的上傳機會。

此外我們也有用到keras的ImageDataGenerator，參數的部分參考下圖，旋轉、水平與鉛直移動是最主要的，只要注意不要動到水平鉛質翻轉的參數即可，這個API最主要的目的就是增加樣本的泛化程度，對應至招牌及拍攝傾斜的角度很實用。

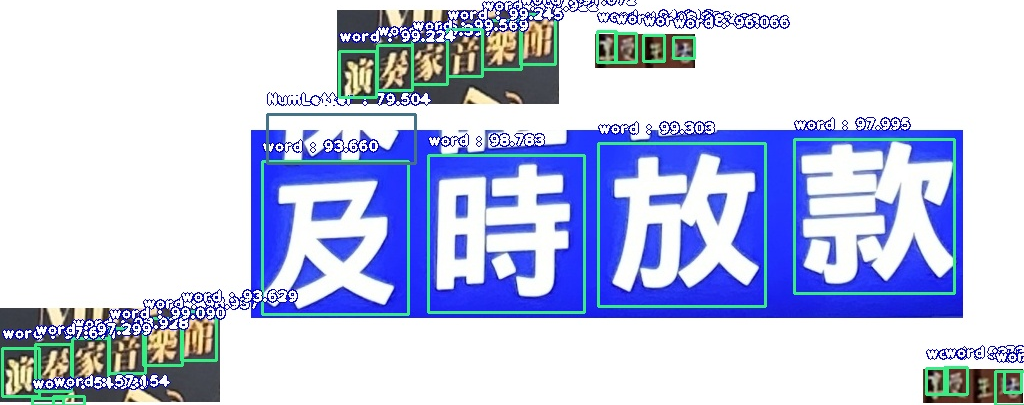


1. 分析與結論
2. 分析

首先是前處理補底的地方，是能最明顯看到效果差異的部分，單只是補白色底前後就有很大的偵測差異，推估可能跟照片尺寸以及Anchor box設計的問題，其次就是閾值的設定，當我們設定太小時，會出現太多環境雜訊被當成字，因此前處理都會統一補上白色底並且置於正中間(另一個實驗發現當物件在圖片的邊緣也有偵測效果不佳的問題)，並在偵測時調整閾值，以本次比賽最後為例，我們是設定在0.5：

|  |  |  |
| --- | --- | --- |
|  | 補底前 | 補底後 |
| Probability=70 | C:\Users\brian\Desktop\場景文字\ImageAI\img_6000_0002 (2).jpg |  |
| Probability=10 | C:\Users\brian\Desktop\場景文字\ImageAI\out.jpg |  |

同物件放置在圖片不同位置所產生的偵測差異，將圖片放大成右圖，可以看到左下角的圖片多偵測了一些雜訊，而右下角的享受生活則是少偵測到生這個字。這一個舉動在public成績中大概Precision會提升0.03以上，搭配上偵測閾值的效果會更佳，最後我們組閾值是設定在0.5。

針對偵測這塊，依然有很多不足的地方，我們有自己去標記Public資料的正解標籤，而根據錯誤統計會發現，在對Public做偵測的時候，原本空白的題目有4973題，但我們卻偵測出5442題，有正確判斷空白的只有4821題，因此其他的錯誤雖然在分數計算上不影響Precision，但卻對1NED的部分有大幅度的失誤(因為把621題有字的題目給當成空白了)，改進的部分我們留到最後在說。

而對於分類的部分，所有的縮放處理都使用到CUBIC，這是一種可以得到較平滑的縮放插值算法，特別是用在放大的部分，在我們內部TestData測試時，還未補齊所有類別資料至一定數量前，準確度其實都能超過90%以上，並根據以前做過的手寫字辨識，在充足資料的狀況下(每個分類至少50個樣本)準確度可以突破95%，因此我們認為模型上的改進對於我們所在的分數階段不會是首選。分析之後，真正扣分的錯誤一部分是前面的偵測模型只要認為是空白，就不會進到分類，這部分會影響1NED至少0.1的分數；另一部分則是源自於訓練樣本的不足，在我們統計下，Public總共有1685個字類別出現，而我們的1111還漏掉1022個類別，這些類別總出現次數占約10%，因此這部分也扣了1NED至少0.1的分數，加上模型本身的錯誤率，最終我們組在Public得分是0.673、Private則是0.666。

1. 改進

首先針對偵測部分，第一是增加偵測的樣本，我們發現模型對於像素低或者物件極小的樣本會有較大的錯誤率；第二個是對所有的訓練偵測樣本做正規劃，最主要是對圖片尺寸做，因為我們ImageAI的Anchor Box是用k-mean計算，很容易被圖片大小的例外值誤導(上至幾千像素，下至不到二十)；第三是改變ImageAI模型的參數\_\_train\_ignore\_treshold，這參數會自動刪去如果真實物件跟Anchor Box計算之間的差異太大，會將此訓練樣本丟棄，這可能也是為什麼我們對小物件的效果特別差；最後則是繼續進行Probability threshold的參數調整，門檻越高precision也會越高，但是就會導致少偵測出字元做分類，因此這部分也是可以繼續研究的。

分類的部分最大的問題也是出在樣本數量，第一要解決的是樣本數極度不平均的問題，原本Train資料切出來的最多一個分類有幾百筆，最少則只有一筆，而經過挑選我們只選了超過四筆資料的進行訓練；第二則是要補充更多的分類資料量，盡可能的覆蓋題目出現的字；第三則是如果前兩者工作做足，則可以考慮最後的改進方法，去選擇另一個更佳的分類模型，或考慮多個分類模型多數決。

硬體也是一個問題，會推薦如果要使用ImageAI可能會需要更加強大的硬體，因為其吃效能的程度很高，三千多張偵測照片的訓練就一圈就需要兩小時(batch size = 4, GPU為1060 6G跟K80)，如果硬體不能換，那就要考慮不要使用ImageAI而使用其他YoloV3的套件，而兩階段的模型架構在跑輸出的時候也需要很長時間，以Public來說我至少需要30分鐘才能得到CSV的結果檔案上傳，而Private三倍的題目則需要超過一小時半。

1. 其他假想

由於時間關係有些理論上的想法未能嘗試：

* + 1. 多設計一層分類器

在圖片進到偵測繁體字之前，先設計一個判斷圖片中有沒有字的分類器。優點是可以讓每個模型的分工更明確，個別準確度提升，但缺點是這樣的架構通常是乘法關係，0.9\*0.9比0.92\*0.92\*0.92要更高分。

* + 1. 圖片改用灰階圖

在分類模型的訓練及判斷時，圖片不採用RGB彩圖以達到降低噪點的功能，且灰階圖可以進行如侵蝕、膨脹等圖像處理，來讓圖片單純化。

* + 1. 對中文字做分流

試圖去降低分類的負擔，譬如先做一次分群(以非監督為主)，分群可能會先做一次大的分類，例如同部首或相近字，然後再丟到各自的分類模型，但這可能也需要對文字理解有造詣的專家，才容易實現；或者就是去訓練新的非監督式模型。

1. 程式碼

程式碼另外附在MainFold，其中包含資料處理、訓練流程、預測等。且有附完整README.md檔案交代安裝配置環境及說明。

1. 使用的外部資源與參考文獻

Moses and John Olafenwa(2018--, Mar). ImageAI, an open source python library built to empower developers to build applications and

systems with self-contained Computer Vision capabilities.

<https://github.com/OlafenwaMoses/ImageAI>

Chollet, Francois(2015). Keras. <https://github.com/fchollet/keras>. GitHub repository.

李謦伊(2020年11月5日)。Inception 系列 — InceptionV4, Inception-ResNet-v1,

Inception-ResNet-v2。<https://medium.com/ching-i/inception-%E7%B3%BB%E5%88%97-inceptionv4-inception-resnet-v1-inception-resnet-v2-42be5d23b2ec>

G. T. Wang(2018年9月18日)。Keras 以 ResNet-50 預訓練模型建立狗與貓辨識

程式。<https://blog.gtwang.org/programming/keras-resnet-50-pre-trained-model-build-dogs-cats-image-classification-system/>

**聯絡資料**

* **隊伍**

|  |  |  |
| --- | --- | --- |
| 隊伍名稱 | Private leaderboard成績 | Private leaderboard名次 |
| 這就是我 | 0.665607 | 19 |

* **隊員(隊長請填第一位)**

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名(中英皆需填寫) | 學校系所 | 電話 | E-mail |
| 蔣明憲 (Ming-Hsien Chiang) | 政治大學資訊科學研究所 | 0936267686 | brian295639@gamil.com |
| 林冠霆（Kuan-Ting Lin) | 長庚大學資訊管理學系 | 0908020792 | a029802184@gmail.com |
| 黃玥菱 (Huang-Yue Ling) | 長庚大學資訊管理學系 | 0956390928 | mia910131@gmail.com |
| 唐碩謙 (Shuo-Chien Tang) | 長庚大學資訊管理學系 | 0903628789 | hubert112247@gmail.com |
| 蕭靖騰 (Ching-Teng Hsiao) | 長庚大學資訊工程學系 | 0925898901 | tengeffort@gmail.com |

* **指導教授**

若為「連結課程」的課堂作業或期末專題，請填授課教師，以利依連結課程彙整。

若非「連結課程」，但有教授實際參與指導，請填寫該位教授。

若以上兩者皆非，可不予填寫。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 教授姓名 | 課程 | 課號 | 學校系所 | E-mail |
|  |  |  |  |  |

1. 官網文件<https://github.com/OlafenwaMoses/ImageAI/blob/master/imageai/Detection/Custom/CUSTOMDETECTIONTRAINING.md> [↑](#footnote-ref-1)