1. 方法描述 - 演算法原理與實作方式描述

所有的圖片讀入後，會先將其resize成224\*224，並將值normalize為0-255區間，因為後續有些方式必須在整數的型態中才能使用。

1. Raw image：

使用原始圖片進行辨識，同時使用到RGB 3個channel的資訊，使用SSD的方式求出與train dataset最小誤差之圖片即將改圖片歸為該類。

1. HOG：

方向梯度直方圖是用於目標檢測的特徵描述器。這項技術是用來計算局部圖像梯度的方向信息的統計值。HOG描述器最重要的思想是：在一副圖像中，局部目標的表象和形狀（appearance and shape）能夠被梯度或邊緣的方向密度分布很好地描述。

實作上使用skimage中的hog函式，此函式是使用灰階的圖片做運算，因此必須將彩色圖片轉為灰階後再使用。使用的參數如下：orientations=9, pixels\_per\_cell=(8, 8), cells\_per\_block=(8, 8)。完成後再利用結果hog\_image做SSD來求出其分類。

1. Local Binary Pattern：

局部二值模式在紋理分類問題上是一個非常強大的。其是一個簡單但非常有效的紋理運算。它將各個像素與其附近的像素進行比較，並把結果保存為二進位數。LBP最重要的屬性是對諸如光照變化等造成的灰度變化的強健性。

實作上是使用skimage中的local\_binary\_pattern函式，此函式是使用灰階的圖片做運算，因此必須將彩色圖片轉為灰階後再使用。使用該方式時，將採點數P設為24、半徑r設為3。求得其每個pixel的Binary Pattern之後，再利用其做SSD來求出其分類。

1. Co-occurrence Matrix：

由於紋理是由灰度分佈在空間位置上反覆出現而形成的，因而在影象空間中相隔某距離的兩畫素之間會存在一定的灰度關係，即影象中灰度的空間相關特性。灰度共生矩陣就是一種通過研究灰度的空間相關特性來描述紋理的常用方法。

實作上是使用skimage中的greycomatrix函式，此函式是使用灰階的圖片做運算，因此必須將彩色圖片轉為灰階後再使用。使用該方式時將d設為(1,1)，求得了Co-occurrence Matrix之後再利用其做SSD來求出其分類。

1. Color histogram：

將各個channel的數值分成256個bins統計數量（也就是剛好一個數值一個bin），接著再利用每張圖片中，256\*3個features做SSD來求出其分類。

實作上是使用opencv中的calcHist函式來統計出各個channel的Color histogram。

1. Gabor Filters：

Gabor Filters，是一種用於紋理分析的線性濾波器，即其主要分析的是，圖像在某一特定區域的特定方向上是否有特定的頻率內容。研究發現，Gabor Filters適合於紋理表示和辨別。在空間域中，2D Gabor Filters是由正弦平面波的高斯核函數。

實作上是使用skimage中的gabor\_kernel函式來做出不同的2D Gabor filter，利用相異之frequency、theta、sigma\_x、sigma\_y來做出filter後，再以此與原圖做convolution。得到結果圖後，再將每張結果圖求mean和var當做feature vector，完成之後將結果利用SSD來求出其分類。（之所以要求mean和var當做feature vector，是因為拿convolution後的結果去做運算的話，每張圖片共會有224\*224\*8個值，導致運算量過於龐大。）

1. Bag-of-Features（Bag-of-Word）：

建立BoW模型主要分為如下幾個步驟:

* 1. 特徵提取

假設有N張圖片，第i張圖片可由n(i)個image patch組成，也就是可以由n(i)個特徵向量表達，特徵向量可以根據特徵問題自行設計, 常用特徵有Color histogram, SIFT, LBP等。

* 1. 生成codebook

對上一步得到的特徵向量進行分群(可以使用K-means等分群方法)，得到K個中心，用各群中心構建codebook。

* 1. 根據字典生成直方圖

對每張圖片計算該圖片的每個「單詞」應該屬於codebook中的「哪一類」單詞，從而得到該圖片對應於該codebook的BoW表示。

實作上在特徵提取這一步驟使用了SIFT來做提取，該方法因為有專利的關係無法在一般的opencv中使用，必須安裝opencv-contrib-python==3.3.0.10。

以validation的檔案為例，train dataset總共可以提取出467224個維度128的特徵。接著再將其使用scipy套件提供的k-mean分成512個類別後，再將test dataset中的每張圖片分別歸類後做成histogram。完成之後使用train data和test data的histogram，利用SSD求出分類。

1. 實驗結果

|  |  |  |
| --- | --- | --- |
|  | Validation | Test on Kaggle |
| Raw image | 45.70% | 52.46% |
| HOG | 20.47% | 22.54% |
| Local Binary Pattern | 19.21% | 14.48% |
| Co-occurrence Matrix | 39.68% | 35.89% |
| Color histogram | 57.37% | 62.59% |
| Gabor Filters | 17.44% | 13.10% |
| Bag-of-Features | 15.54% | 14.11% |

1. 結果討論 - 對於實驗結果的⼀些解釋和討論

以結果上來說，基本上準確率普遍都是偏低的，表現最好的選項有點超乎意料之外是Color histogram，在kaggle上面的結果還有超過60%。至於其他大部分的方法都低於25%。我覺得有個很大的問題是這次的實驗都是直接把輸出的特徵利用SSD來求取類別，但從網路上找到的相關資料都是要在搭配一些傳統機器學習的方法做分類（如：SVM）。或許以後可以多再搭配其他各類型機器學習的方式，看看結果是否會比較好。

另外以HOG方法為例，如果把其特徵圖畫出來可以發現，他在擷取特徵上出現了些許偏差，很多圖片中他抓到的並不是草的輪廓，而是背景上黑白相間的部分，然而背景並不應該是用來被判斷種類的依據，或許這也是其中一個會讓結果不如預期高的原因。

只是Color histogram在本次的結果是最佳的還是有點讓人驚訝，畢竟每張圖片都是有植物和背景，植物的綠色分佈應該是不會差異太大才是。電腦視覺的領域有點時候或許不像我們想像的那麼直觀。

1. 問題討論 - 作業撰寫中遭遇的演算法問題與實作的困難

我覺得最大的困難還是在最剛開始的時候對各個演算法的不熟悉，雖然有了函式的協助，但很多時候參數該如何設定並不是很直覺，所以導致在寫每個演算法的過程中花了很多的時間在釐清演算法的原理上面。不過在寫完這次的作業之後，把每個方法的結果feature都嘗試以圖片的方式呈現出來，對於各個演算法背後的概念也有了更深入的了解。

雖然這次的結果並不是到很高（原因及改進部分如前所述），但懂了這些方法之後，以後在做圖片資料的分析時，一定會有很好的幫助。

另外在寫程式過程中遇到最大的麻煩是SIFT這個演算法目前是有專利的，因此在運用上並不是那麼自由，opencv也將其從原本的函式庫中移除，必須要安裝特定版本的函式庫才能使用。

*pip install --user opencv-contrib-python==3.3.0.10*