Flower Detection and Classification

Author: Zhili Xu

Date: November 21st, 2018

Email: zhili.xu@mail.utoronto.ca

Teammate: Yanhan Wen

Abstract

The objective of this paper is to demonstrate the flower detection and classification tools and methods that we used to detect and identify the flower species in an image. We use Tensorflow and dataset from Kaggle to train our custom flower detector and then use '17 flower' dataset from Oxford Flower Species Recognition to cluster and identity the input flower species.

1. Introduction

Object detection is always a popular topic in computer vision, there are many quality models to detect objects such as cars, persons, traffic-lights. However, upon researching, most popular object detection models are not trained to detect flower from images. It is understandable that flower detection is not the priority in object detection area, because, firstly, flower is relatively small compares to other objects on the street, secondly, flowers detection is not as profitable as other object detection research area such as auto-driving and facial recognition.

Although flower detection is not the priority for computer vision research group, what innovate us to create a flower detection tool is difficulties from another course that we are taking. From that course, we were required to identify flowers species, and during that, we found that there are many limitations on most plants' identification tools out on the public. Most of them require user to input a nearly perfect image from the flower, which the flower fills the entire image. We are hoping to improve this process by detecting and segmenting the user input image, which finds the flower location on the image, thus users does not require to submit a perfect flower image.

During the researching process, we found a useful paper from Oxford Flower Species Recognition and we are using it as a base to develop our flower detection tool[1]. The paper use 'BiCoS' as its segmentation method to extract the foreground of the image. However, it does not solve the problem which identify the location of the flower if it does not fill the image. Thus, this is the main objective that we are hoping to solve, which detect the flower location from an image, crop the part from the original image, and use it as an input for later identifying process.

The dataset, we are using to train in order to detect flower, is an open source flower dataset from Kaggle[2] which contains about 200 images. We are expecting correctly detect the flower location if there is only one flower in a reasonably image. However, we are also expecting a large amount of false-positive on the background of the image. This is because the dataset also contains background with leaves and grass. Therefore, when a poorly-taken image is presented, for example, a flower in the middle of field [Fig.1], may return false-positive detection score.

Fig.1 A yellow flower on the field

2. Method

The process of identifying the flower species breaks into two main part. The first part requires detecting the bounding box of the flower in the images, and save the cropped images from the bounding box. The second part uses the saved cropped images as inputs and classify the flower species. The process is developed by a group of two. Zhili Xu, author of this report, is responsible on the implementation of the first part, flower detection. Yanhan Wen, author's teammate, is charged for implementation the second part, flower recognition. The operating environment is on Windows 10 with CUDA 9.0 GPU support.

2.1 Flower location detection

The flower detection can be separate into 4 process. We start with setting up the Tensorflow API environment which will be using to train our detection model. We then introduce how to generating the training data. After that, we demonstrate how to train the model using the dataset and how to create the bounding boxes with the trained model.

a) Setting up tensorflow model environment

We are using Tensorflow API to train our flower detector model [3]. We are following the installation page on Tensorflow github page. After properly setting the Tensorflow, we can start training our dataset. Our dataset is from Kaggle[2] which contains 210 flower images with 128 * 128 pixel size. We placed the dataset in the image folder under Tensorflow's "objection_detection". Then the dataset is separated into train and test folder with 80 versus 20 ratio.

b) Generating Training data

After properly storing the dataset images, we are using the "writetocsv.py" script to label each image. Labelling is an important step in object detection, as the object detector needs to understand what class it is detecting. The script generates two separate label files, one for each train and test images. The label file contains the size of image, the class of the object, and the location of the object for each file. We then convert the two label files from csv type into TFRecord file type, which serve as input data to Tensorflow training model and will be used to train flower detection classifier. The script is implemented in "generate_tfrecord.py" [4].

c) Training the data

Before using the Tensorflow API to train our data, we need to create a label map and modify the training configure file. Tensorflow API reads the label map on what each object is by mapping the class name and class id number. The "labelmap.pbtxt" is placed under the training folder for later use. After that, we will need to configure the training pipeline. We select the Tensorflow "faster_rcnn_inception_v2" as our model, because it is one of the best accurate object detection models. R-CNN generate a set of proposals for the bounding box and run the images in the bounding box with an SVM to detect what object in the bounding box is. Faster R-CNN reuses the region proposal that was calculated from forward pass of CNN, so that CNN is used to both classification and region proposal. Therefore, only one CNN is trained, which speeds up the training process [5]. We then modify the parameter of the "faster_rcnn_inception_v2.config" file to match with our training data and labels. After all the configuration is completed, we are using the "train.py" script in the legacy folder to generate our classifier. The loss value of the training process starts at 2.5 and after 11413 steps, the loss value plateaus and is less than 0.01, thus, the model is well-trained and we stop the process.

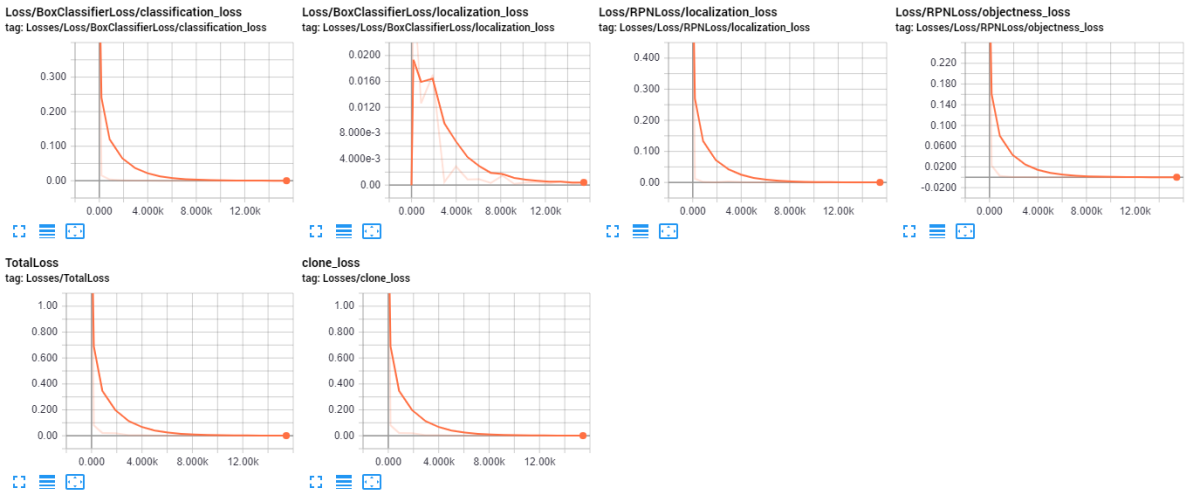

Fig2 The loss function graph produce by Tensorboard.

d) Detection and Crop Image

We generate the frozen inference graph from our newly trained model, which will be used as the object detection classifier. We use the frozen inference graph in the "flower_detection.py" script to detect the location of flower in the image. The script is modify from the "objection_detection_tutorial.ipynb" file, which use our own flower detection classifier and modify the bounding box to accurately detect the location of the flower. The part inside the bounding box is cropped and saved for future flower species identification use.

2.2 Flower Species Identification

In the beginning of the implementation of species recognition, we decided to use SIFT detection to match the query image with the dataset. However, SIFT detection does not take colour of the image into account. Colours of the flowers are important aspects on identifying its species, thus we change method to logistic regression.

a) Compute feature descriptor:

We are using the last layer of neural network before softmax to compute our feature descriptor. Neural Nets learns the image features in a hierarchical fashion. Lower layers learn feature like corners and edge and higher layers learn feature that representing the object, in our case, flower species. We are using the Keras_Inception_v3 as the pre-trained model, since it is fast and weight size is small. We then use the model to extract feature descriptor and labels on the oxford 17 flower dataset[6].

b) Build the model

We will use the features and labels to train to recognize the flower species. We separate the dataset into train and test image categories, and then using the sciki-learns logistic regression, we are able to train these feature and labels which will be use to identify flower species.

c) Predict query image

For the last step, the program is using the trained model and the cropped image from previously saved flower detection image to predict which category the query flower belongs to.

3. Result and Discussion

3.1 The Dataset

During the evaluation of the program, we are interested in two accuracy measurement: the flower detection accuracy and species recognition accuracy. The dataset that is used for testing accuracy contains a variety of flower pictures, taken by us or found online. The numbers and size of the flowers in the images are vary, and the flower species belong to the "Oxford17" flower species that was used for training. It is important to note that, although each image in the dataset contains different number of flowers, but flowers in an image all belong to the same species.

3.2 Flower Detection Result

The "flower_detection_test.py" script is used to test the flower detection accuracy, and the test result is promising. The flower detection program has 100% accuracy on finding the bounding box of the flower location in the image. The only case which the detection program performs under the expectation is when the flower fills the entire image. In this case, the bounding box only contains parts of the flower, however upon further testing, the species recognition result is not affected by only providing parts of the flower. This is because the parts of the flower are the key features of that species, so that the recognition program can still identify which species it belongs to.

Fig3. Sample results from flower detection program. The part inside the bounding box is saved for later species identification use.

3.3 Species Identification Result

The “test_accuacy.py” script is used to test species recognition accuracy. Used to cropped imaged from previous step, the program predicts and outputs top 5 flower species that the query flower belongs to. The program achieves 82% top 1 prediction accuracy, which means the query flower species matches with the first slot of the prediction species. Besides, it also achieves 96% top 5 prediction accuracy—the query flower belongs one of the top 5 prediction. Before the experiment, we expect the program is not able to predict the images which contains multiple number of flowers. However, after “flower_detection” script cropping the image, the program is able to correctly identify it.

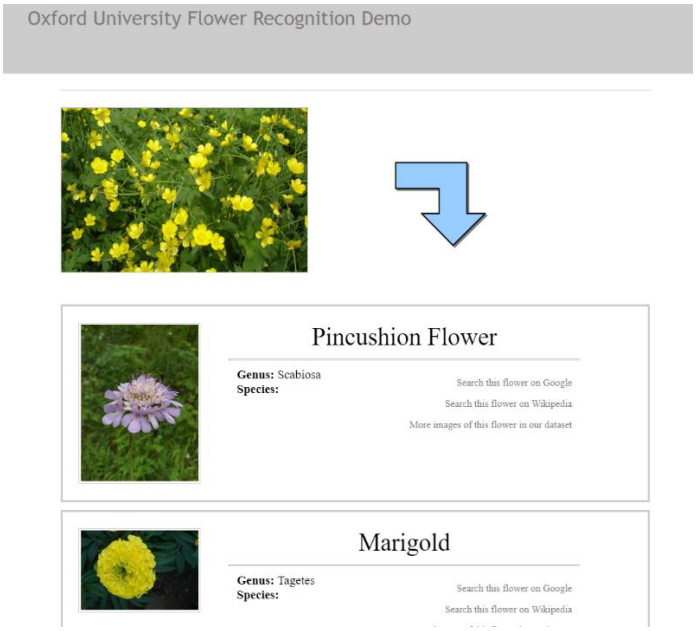

Fig4. Oxford Flower Recognition Program is not able to identify pictures contains multiple buttercup flowers

Fig5. Our program successfully identifies it.

3.4 Additional Test

After the main test is completed, we are also interested in the effect of providing a cropped image, that is how much does the photo produced by flower_detection help the classification process. Therefore, we omit the flower detection process and only run the prediction script on the dataset. The test results show that, the top 1 prediction and top 5 prediction accuracy is, 70% and 79% respectively

4. Main Challenges

One of the challenges when developing the program is the lack of dataset available. We originally decided to use oxford "flower17" dataset for both flower detection and recognition training. However, this dataset contains too much background noise. Thus, we decided to use another dataset such as the flower fills the entire image, so we can use script to create image labelling. Furthermore, the size of dataset is smaller than the ideal dataset size. In order to train an object detection model, the size of the dataset should be larger than 500. However, the dataset of the flower detection only has 210 images and dataset used for flower recognition only has 80 images for each species.

Another challenge is dealing with multiple bounding box. Even after tuning the non-maximal suppression parameter in the Faster R-CNN training model, there are still overlapping on the bounding boxes. After observing the location of the bounding boxes, we find that all the bounding boxes contains the flower in it. Therefore, we pick the smallest bounding boxes to use for our flower detected location and use it for the flower classification process.

5. Conclusion and Future Work

The flower detection and recognition program uses Tensorflow API for locating the flower in an image and use Keras logistic regression to identify which species it belongs to. The program accomplishes our main objective, which is able to identify 17 flower species in an image that is not well-taken and it achieves 96% of top 5 prediction accuracy.

For the future development, we are planning to increase the data size for both flower detection and species recognition, which should improve our results.

Reference

[1] Y. Chai, V. Lempitsky and A. Zisserman, "BiCoS: A Bi-level Co-Segmentation Method for Image Classification," 2011. Available: http://www.robots.ox.ac.uk/~vgg/publications-new/Public/2011/Chai11/chai11.pdf.

[2] O. Belitskaya, “The Dataset of Flower Images”, 2018. Available: https://www.kaggle.com/olgabelitskaya/the-dataset-of-flower-images

[3] “TensorFlow Models” Available: https://github.com/tensorflow/models

[4] “Raccoon Detector Dataset” Available: https://github.com/datitran/raccoon_dataset

[5] D. Parthasarathy, “A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN”. Available: https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

[6] M. Nilsback, A. Zisserman, “17 Category Flower Dataset”. Available: http://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html