**Flower Species Identification Project Report**

Yanhan Wen

Student number: 1001366782

UtorID: wenyanha

Email: [yanhan.wen@mail.utoronto.ca](mailto:yanhan.wen@mail.utoronto.ca)

Teammate: Zhili Xu

**1. Introduction**

The problem we are trying to solve is that we want to implement a program that takes an image with a flower and report top 5 most likely species that this flower belongs to with sample picture of each species. The input image doesn’t have to be a close shot to the flower. Our program should be able to identify where the flower is in the image and use a classifier to classify the segment that contains the flower.

What inspires the problem is from an assignment of a course we took this semester. The assignment is about taking pictures and identifying blooming flower’s species in GTA. It is inefficient to read the instruction of identifying flower and carefully inspect every flower. Using the technique we learned from image understanding course comes to mind.

We found that the Visual Geometry Group from University of Oxford already implemented a flower species recognition program. The program takes an image of close-up shot of a flower and predict what species it is. They have a paper documented how they completed the project [1]. In general, they used a method called GrabCut to segment the flower from the background in training image. Then, for the recognition part, they used technique of concatenating the Bag-Of-Words histograms of Lab color and SIFT descriptors. The program they implemented can achieve 80% accuracy on identifying flower species. Conveniently, the project provided flower dataset they used for training on the website [2]. The smaller dataset has 17 species with 80 pictures of each and the larger dataset has 102 species with 80 pictures of each. We utilized these datasets in our project.

What we expect to work is that given an image that contains a close-up shot of the flower, our program can identify the species. Given an image of flower but not necessarily a close-up shot, our program should identify the species of the flower. Given an image containing multiple flowers within the same species, the program should return what species these flowers are. The limit of existing method is that they perform well with a close-up single flower photo. They don’t give correct result when the flower doesn’t take most of the space in an image or when there are multiple same species flowers. This is the challenge we need to face as well. We want to first use object detection techniques to identify flower in an image and then using convolutional neural network and linear regression to recognize species of the flower in an image. The goal is to achieve high accuracy in both identifying flower in image and recognizing species to reach an overall high accuracy at the end.

**2. Method**

Our high-level approach to the problem is first segment the flower from the input image and then recognize the segmented flower. This naturally divides the problem into two parts. Zhili Xu is responsible for capturing the flower segment from an image that contains flower and Yanhan Wen is responsible for identifying the species of the flower given the segmented flower.

**a. Segmentation of the flower from the input image**

We utilized Tensorflow’s object detection api, and there are 4 major steps in solving the problem.

1) set up TensorFlow environment

2) gather flower dataset for segmentation training

We wanted to train a model that takes an image containing flowers and then segment the area in the image that contains flowers. The dataset we used to train is Kaggle’s The Dataset of Flower Images that contains 210 flowers image [10]. The API requires to label where the target object is in an image. We used file wrtietocsv.py to read all training data and labels and label where the flower is in the image. Since the image in datasets are all close-up shot of flower, we label the entire image as the flower. Then we used generate\_tfrecord.py file to generate a record file contains training image’s label that TensorFlow’s object detection api can read.

3) train

We used TensorFlow’s object detection api [12] to model our dataset and labels. The model is trained using a convolutional neural network named as Faster RCNN Inception V2 Pets. The architecture is built based on Inception Resnet V2. We stopped training at step 14413 because the loss is already small. After training, we stored the model in a file for future use.

4) crop

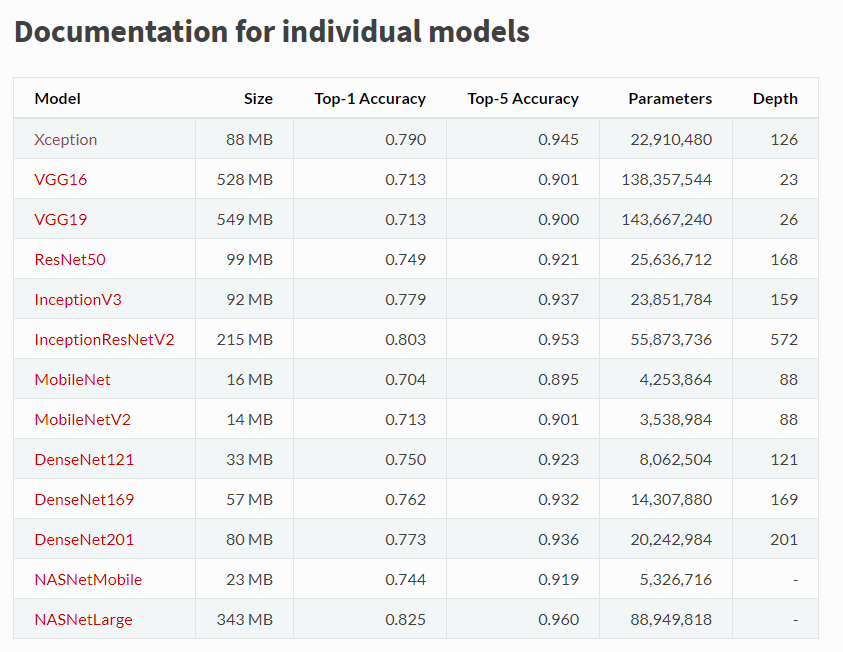
Given an image, the trained model will circle regions that contains flower. After non-maximal suppression, there are still overlaps between regions. Non-maximal suppression eliminates points that do not belong to the critical area. It makes sure that a particular object is identified only once. We decided to use the smallest region because all the region’s contain flower and the smallest region can give the most information about the flower in it.

**b. Identify flower species**

We will use Oxford’s flower dataset with 17 flower species as our training data [2].

1) compute feature descriptor

We decided to use the last layer before softmax function of a trained convolutional neural network as our feature descriptor. There are several deep neural networks that is public in Keras so that we used one of the models to compute our feature descriptor for both training and query image. Kera is a neural network library that is built on top of TensorFlow. How convolutional neural network works is that it takes an image and apply filters to the image layer by layer to reduce the size of the input and keep the important features. At the end, there will be several fully connected layers that classify the input image [4]. What we want here is the reduced image with features kept from the input image before the classification step. There are several convolutional neural networks available in Keras such that their weights are trained on ImageNet [5]. Here is difference chart [5]:



We decided to use InceptionV3 as the network to compute feature descriptor because it has relatively high accuracy with small size. From Keras Model API [6], [7] and [8], we got a model that can fit a data and predict the layer before softmax function. This is shown in line 18 and 19 in the file extract\_feature.py. What’s left is to compute descriptor of every training image from Oxford flower dataset. This is shown in line 26 to 46 in extract\_feature.py. Note that InceptionV3 model takes input image of dimesion 299x299x3. We used kera’s image.load\_img function to load training data with specific size and convert it to numpy array. The dense feature descriptor of an image using this model is of size 1x131072. We used h5py to save the descriptors into h5 files to convert back to numpy array easily in later use.

2) build the model

The model is trained in file train.py. The dataset we use is the 17 flower species data from Oxford. We read the features descriptors and labels computed in previous step from file, convert them to numpy array and split them into train and test data using sklearn’s split function from line 21 to 35.

Since each descriptor is labeled, we can use linear regression to classify all the descriptors. Since linear regression give too much weight to data far from the decision frontier, logistic regression should be used to fit the data [9]. Logistic regression uses an activation function to limit the output from linear regression to categorize output better. The model is train in line 38 to 39 using sklearn’s LogisticRegression model [9]. Line 45 to 76 compute accuracy of the prediction on test dataset. On test dataset, the most likely match of the query flower has 91.91% accuracy. Top 4 most likely matches contain the species has 98.90% accuracy and top 5 most likely matches contain the species of the query flower with 100% accuracy. The accuracy is 10% higher than the Oxford’s project by using pretrained neural network.

3) predict the query image

The predict function is stored in predict.py. It is similar to calculating test accuracy. It reads the serialized trained model from previous step in line 19. It computes the feature descriptor of the query image using function get\_des(im) from line 48 to 60. Line 25 to 45 uses the model to predict on query image and displays top 5 species that match query image along with species’ sample image for user to pick manually since top 5 result will include the correct species.

**3. Results and Discussions**

In the early stage of approaching the problem of identifying species, I wanted to compute SIFT descriptor of the query image and every training data from oxford’s flower dataset [2]. The species that contains the most matches will be the flower’s most likely species. What SIFT does is that it is an algorithm that extracts features points from the image that are invariant to scale. It could be used to find matches between two images with different scale, rotation, viewpoint, etc. However, there are two constrains of this approach. First, it is inefficient. We will need to compute 1360 images’ SIFT descriptor and compute similarity score one by one with query image. Second, SIFT descriptor only works on greyscale images. Some flowers have similar shape but different color such as bluebell and tulip. In this case, our program can’t distinguish these two species.

First approach left me two problems to solve. First, compute a feature descriptor that includes color information of the image. Second, how to classify flower species efficiently. From research and inspiration of using pretrained model from assignment 5, I decided to use the last layer before softmax function of a trained convolutional neural network as our feature descriptor. Using this technique, I can utilize powerful and high-accuracy neural network existed to compute feature descriptor efficiently [3]. This helps keep the color information of the image in the descriptor and each image is corresponding to only one descriptor. This method helps retain more information and computes less but accurate descriptors comparing with SIFT method.

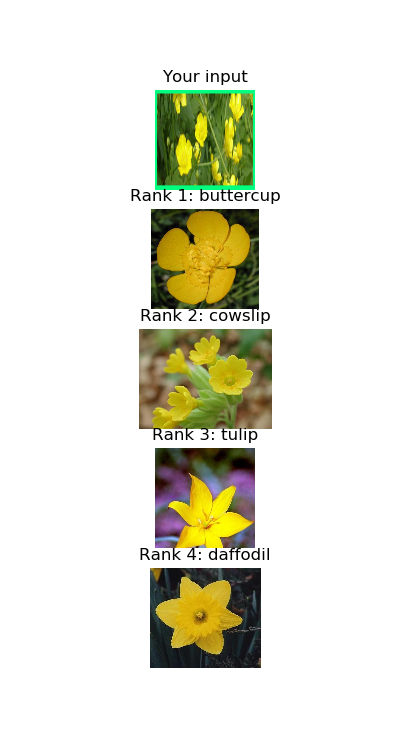
At first, we wanted to use KMeans to learn the data because it is faster than fitting the model. It is commented out from line 42 to 43 in train.py. KMeans technique clusters similar data into same group and then computes their means. When making prediction on query image, it computes distance of query image to every cluster mean and pick the one that is most similar to it as the result. We used sklearn’s NearestCentroid model to compute cluster mean for each group of data. From 20% test data (272 images), I achieved 77% accuracy. It is not a bad result, but we want to improve on that to reach higher accuracy to compensate the loss in segmentation step. Using logistic regression, we can achieve 91.91% accuracy on test data and top 5 matches will contain the query image’s species with 100% probability. This boosts our accuracy by 14%.

Comparing with flower detection program provided by University of Oxford’s Visual Geometry Group, our program performs 10% more accurate given close-up shot of a single flower image. Moreover, combining with segmentation step to pick out flower from an image done by Zhili Xu, the most likely species identified by our program can achieve 82% accuracy and 96% accuracy on identifying correct species in top 5 most likely result. The test set we used has 5 pictures for each species with close-up picture, multiple flowers, and far from flower picture mixed up together. This is something the program from University of Oxford couldn’t achieve.

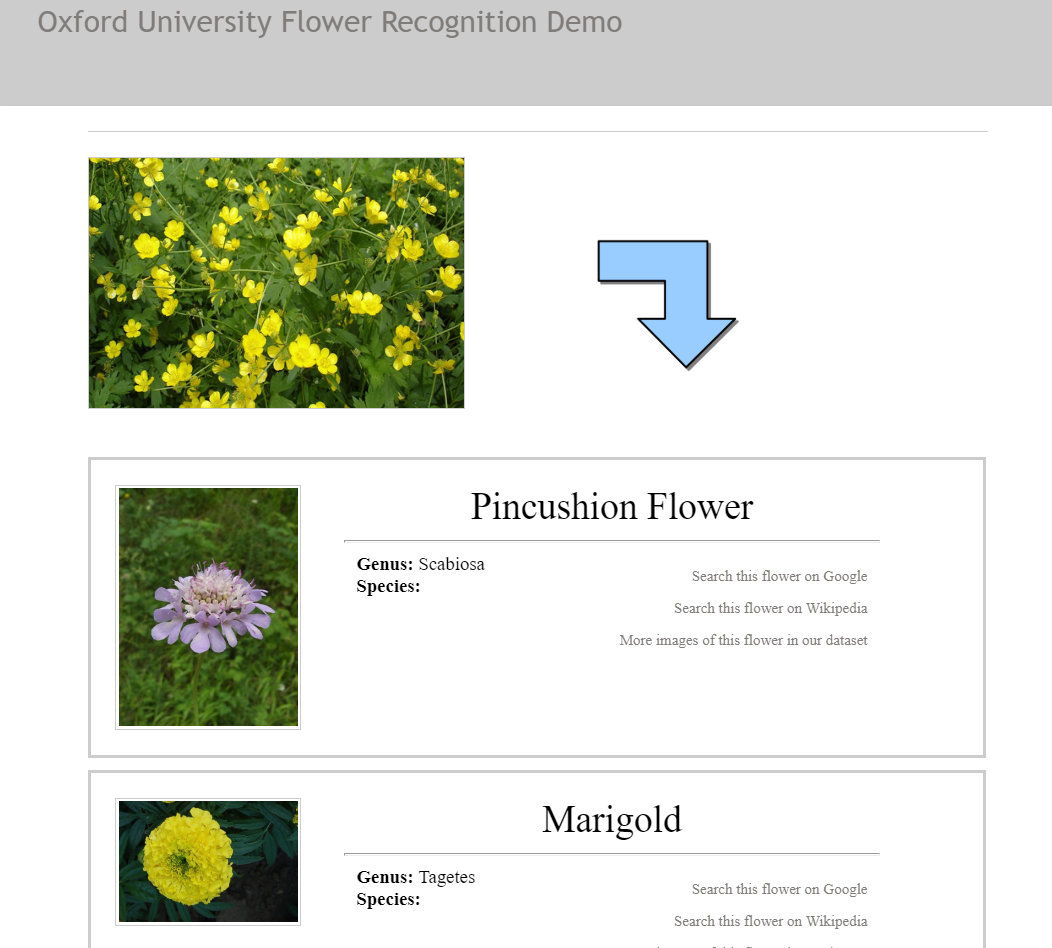
Here is an image that contains multiple buttercup species:



This is the prediction from our program. Note that the image is cropped to focus on one or two flowers:

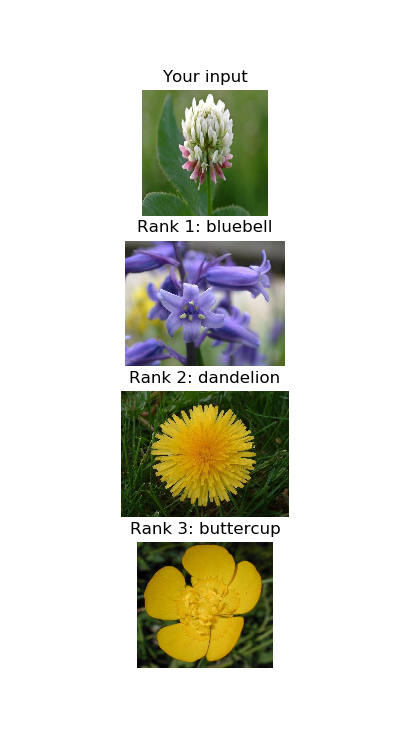


This is the prediction from Oxford’s program using the same image [11]:



Our program gives higher accuracy on images containing multiple flowers or flower doesn’t take big portion in the image.

However, the program will not give correct prediction if the query flower is not from 17 species of the training set:



To solve this problem, we will have to train the prediction model with more diverse flower dataset.

**4. Main Challenges**

One challenge I faced is trying to compute a descriptor of training images. I spent long time experimenting SIFT descriptor to solve the matching problem. I computed SIFT matches of one unseen buttercup species image with all images from buttercup group and bluebell group from training dataset from University of Oxford. I incremented the number of matches together and compare these two numbers. Matches from buttercup group is higher than bluebell group. However, the matches are also really high with group cowslip which only has one more petal than butter cup species and sometimes there are more matches in cowslip group. It turned out using convolutional neural network to compute a single descriptor for the image is a faster and more accurate approach.

Another challenge I face is I can’t pick the reasonable model to fit the training data. At first, I used KMeans model, which didn’t result in high accuracy. After this, I used linear regression model, which gives floating point result that is hard to classify. After some more researches, I used logistic regression to fit the training data.

**5. Conclusion and Future Work**

In conclusion, what we did is take an input image, using a trained model to segment the flower portion of the image. Using pretrained neural network InceptionV3 to compute feature descriptor of training images and query segmented image. Train a logistic regression classifier using University of Oxford’s 17 flower species data. Compute the feature descriptor of the segmentation and pass to this classifier just trained to predict the species of the flower.

What I would do in the future is to use a larger dataset with more species of flower to train the classifier to classify more species of flower. Also, more images should be used in each species to enhance the accuracy of the classifier.

**6. References**

[1] Y. Chai, V. Lempitsky, and A. Zisserman. BiCoS: A Bi-level Co-Segmentation Method for Image Classification. In *ICCV,* 2011.

[2] Flower Datasets from Visual Geometry Group of University of Oxford. <http://www.robots.ox.ac.uk/~vgg/data/flowers/index.html>

[3] Transfer learning. <http://cs231n.github.io/transfer-learning/>

[4] ujjwalkarn. An Intuitive Explanation of Convolutional Neural Networks. In *the data science blog,* 2016.

[5] Keras Applications. <https://keras.io/applications/>

[6] Keras Model. <https://keras.io/models/model/>

[7] F. Yu. A Comprehensive guide to Fine-tuning Deep Learning Models in Keras (Part II). In 2016.

[8] Gogul09. Using Keras Pre-trained Deep Learning models for your own dataset. In 2017.

[9] Supervised learning: predicting an output variable from high-dimensional observations. <https://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html>

[10] The Dataset of Flower Images from Kaggle. <https://www.kaggle.com/olgabelitskaya/the-dataset-of-flower-images>

[11] Oxford University Flower Recognition Demo. <http://zeus.robots.ox.ac.uk/flower_demo/>

[12] TensorFlow Object Detection API. <https://github.com/muserxu/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10>