Individual Component Report

Chenqu Zhao
University of New South Wales
Sydney, Australia
chenqu.zhao@student.unsw.edu.au

I. Introduction

This task is to do binary classification of plants into two plant classes named Arabidopsis and Tobacco, which using several computer vision techniques including image preprocessing, image segmentation, feature detection, feature encoding and machine learning. The train data provided is a collection of leaves images for these two plants. During the implementation, I read lecture slides, some report from Google scholar and related Python module documentation and .

II. Method (Implementation)

A. *Image preprocessing*

After reading in images, to normalize their size for future training and predicting, I resize the images to the same scale with the interpolation method of nearest neighbour, which is recommended by the official OpenCV documentation for shrinking images [1]. Then I apply a median filter to enhance the images [2] and use a sharpening filter with the mask of Laplacian plus addition to sharpen the image [3]. This way the noises are removed, and the main object is clearer.

B. *Image segmentation*

I choose watershed to do segmentation, since that it is an interactive image segmentation method and could label the region of background and foreground with different values. After that, the marker can provide the boundary to separate the leaves and background. With these dividing lines, I can extract the plants for further training. After segmentation, I divide the dataset and label set into train data, train label, test data and test label [4].

C. *Feature detection*

Since that SURF is 3 times faster than SIFT and is good at handling images with blurring and rotation, I choose SURF to detect keypoints and describe local features. I obtain features for each image and store them together in a list [5].

D. *Feature encoding*

The main technique of feature representation in my implementation is Bag of Words (BoW). First, I concatenate the features of images together to generate a 2D array as feature descriptors. Then, I set clusters using k-means clustering and store the corresponding codebook [6]. The cluster centres are a kind of visual words which form the vocabulary that is used to represent an image. Therefore, to let the feature descriptors make sense, each of them is assigned to one visual word with the smallest distance using sklearn.vq function [7]. I use a 2D NumPy array to store the feature descriptor assignment results, which represents the feature histogram, and is ready to be trained.

E. *Machine Learning*

I create a random forest classifier model. Since that it is a Bagging technique, the implementation of randomness avoids overfitting. In addition, it can handle very high dimensional data without dimensionality reduction. Also, the training speed is fast, and the importance of variable can be ranked. After fitting it with the train data feature histogram and train labels, I do the same feature detection and feature encoding operation for test data [8]. Then, predict the test labels using the random forest classifier model and test data feature histogram.

III. Experiment

This section is to explain the experimental setup and the evaluation methods and metrics used.

A. *Image preprocessing*

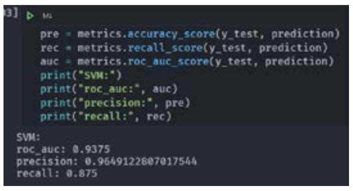
The images are scaled to 255 * 255, since that images with square shape are easier to be preprocessed and trained. I set the median filter size as 5. That is the choice after trying many different odd numbers, which is the best trade-off between noise removal and object clarity. When sharpening images, I use a sharpening filter with the mask of Laplacian plus addition. It is just a simplification of computation, which has the same effects as the Laplacian filter that is good at suppressing noises.

B. *Feature Encoding*

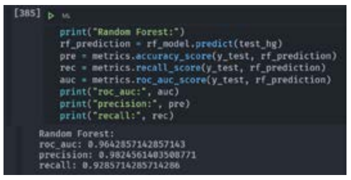
I use 100 as the number of centroids to do k-means clustering. This is chosen from the Elbow method. I iterate the k value from 0 to 150, after each clustering, compute the sum of the squares of the distance from each point to the cluster center to which it belongs. And the sum reaches around 0 when k equals to 100 because each point is the cluster center itself.

C. *Machine learning*

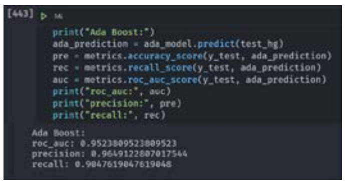
I try with SVM classifier, Ada Boost classifier and random forest classifier and compare their performance. If the number of features is much more than the number of samples, it may cause overfitting in SVM classifier. It may happen that unbalanced data leads to decreased classification accuracy in Ada Boost classifier. Thus, I choose random forest classifier and the figure below shows the performance of these three classifier with default parameters in this project.



```
3]
    pre = metrics.accuracy_score(y_test, prediction)
    rec = metrics.recall_score(y_test, prediction)
    auc = metrics.roc_auc_score(y_test, prediction)
    print("SVM:")
    print("roc_auc:", auc)
    print("precision:", pre)
    print("recall:", rec)

SVM:
roc_auc: 0.9375
precision: 0.9649122807017544
recall: 0.875
```



```
[385]
        print("Random Forest:")
        rf_prediction = rf_model.predict(test_hg)
        pre = metrics.accuracy_score(y_test, rf_prediction)
        rec = metrics.recall_score(y_test, rf_prediction)
        auc = metrics.roc_auc_score(y_test, rf_prediction)
        print("roc_auc:", auc)
        print("precision:", pre)
        print("recall:", rec)

    Random Forest:
    roc_auc: 0.9642857142857143
    precision: 0.9824561403508771
    recall: 0.9285714285714286
```



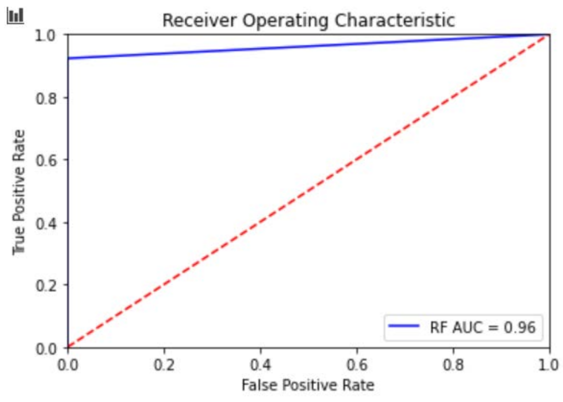
```
[443]
        print("Ada Boost:")
        ada_prediction = ada_model.predict(test_hg)
        pre = metrics.accuracy_score(y_test, ada_prediction)
        rec = metrics.recall_score(y_test, ada_prediction)
        auc = metrics.roc_auc_score(y_test, ada_prediction)
        print("roc_auc:", auc)
        print("precision:", pre)
        print("recall:", rec)

    Ada Boost:
    roc_auc: 0.9523809523809523
    precision: 0.9649122807017544
    recall: 0.9047619047619048
```

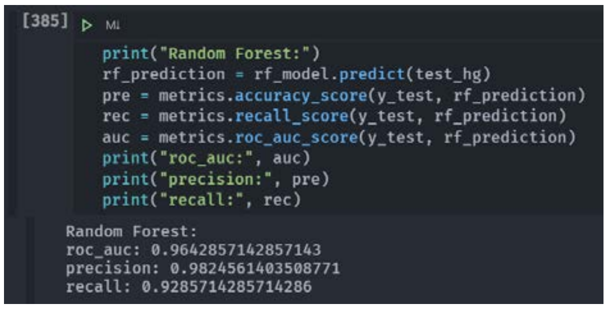
When setting parameters, I choose n_estimators as 41 which is the best value to improve the model accuracy and stability after trying values from 20 to 45. I set max depth as 11 which performs the best classification within a reasonable time complexity.

D. Evaluation Method

I use metrics class from sklearn module to evaluate the classification result. Three of the functions are used, which are to calculate accuracy classification score, recall score and roc score. The accuracy score is the percentage of the set of labels predicted for a sample that exactly match the corresponding set of labels in test labels set. The recall score is the percentage of the true positive samples found out of all positive samples in test labels set. The AUC score is, given a positive sample and a negative sample at random, the probability that the positive sample is output by the classifier is more likely than the probability that the negative sample is output by the classifier. The ROC AUC curve is also implemented for better understanding of evaluation result [9].

IV. Results and discussion





```
[385]
    print("Random Forest:")
    rf_prediction = rf_model.predict(test_hg)
    pre = metrics.accuracy_score(y_test, rf_prediction)
    rec = metrics.recall_score(y_test, rf_prediction)
    auc = metrics.roc_auc_score(y_test, rf_prediction)
    print("roc_auc:", auc)
    print("precision:", pre)
    print("recall:", rec)
```

```
Random Forest:
roc_auc: 0.9642857142857143
precision: 0.9824561403508771
recall: 0.9285714285714286
```

The figure above is the ROC graph of the random forest classifier. As the figure plot, the curve is near the top of the graph, and the classifier reaches the AUC value of 0.96, which is a good enough true positive rate. As the result graph shown, the precision score is around 0.98 and the recall score is around 0.93. Overall, this random forest classifier performs well in this plant image binary classification task.

References

[1] "Geometric Image Transformations," *OpenCV*. [Online]. Available: https://docs.opencv.org/master/da/d54/group__imgproc__transform.html. [Accessed: 04-Nov-2020].

[2] "Smoothing Images," *OpenCV*. [Online]. Available: https://docs.opencv.org/3.4/dc/dd3/tutorial_gausian_median_blur_bilateral_filter.html. [Accessed: 04-Nov-2020].

[3] " Laplacian/Laplacian of Gaussian." [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm. [Accessed: 04-Nov-2020].

[4] "Image Segmentation with Watershed Algorithm," *OpenCV*. [Online]. Available: https://docs.opencv.org/master/d3/db4/tutorial_py_watershed.html. [Accessed: 04-Nov-2020].

[5] "Introduction to SURF (Speeded-Up Robust Features)," OpenCV. [Online]. Available: https://docs.opencv.org/master/df/dd2/tutorial_py_surf_intro.html. [Accessed: 04-Nov-2020].

[6] "scipy.cluster.vq.kmeans," scipy.cluster.vq.kmeans - SciPy v1.5.3 Reference Guide. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.vq.kmeans.html. [Accessed: 04-Nov-2020].

[7] "scipy.cluster.vq.vq," scipy.cluster.vq.vq - SciPy v1.5.3 Reference Guide. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.vq.vq.html. [Accessed: 04-Nov-2020].

[8] " 3.2.4.3.1. sklearn.ensemble.RandomForestClassifier," scikit. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed: 04-Nov-2020].

[9] " sklearn.metrics.accuracy_score," scikit. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html. [Accessed: 04-Nov-2020].