ENHANCEMENT OF FLOWER CLASSIFICATION WITH THE PROFILE FEATURE

[1]*Wei-Hao Fu,* [2]*Kuo-Tsung Tseng,* [3]*Chiou-Yi Hor,* [1]*Chang-Biau Yang*

[1]Department of Computer Science and Engineering,
National Sun Yat-sen University, Kaohsiung, Taiwan
[2]Department of Shipping and Transportation Management
National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan
[3]China Steel Corporation, Kaohsiung, Taiwan
E-mail: cbyang@cse.nsysn.edu.tw

ABSTRACT

In this paper, we propose an elegant method, based on machine learning, for the flower classification. There are three stages in our method. The process begins with segmenting the flower images and removing their backgrounds. Then, we extract the features from the foreground, including color features and texture features. Finally, we train the *SVM* (support vector machine) models and *Adaboost* models with several feature combinations. The experimental material comes from the Oxford-102 category flower dataset. Our proposed feature, named the profile feature, improves about 2% accuracy in the SVM model and about 3% in our ensemble model. It outperforms all models without deep-learning (TensorFlow inception-v3 model). The proposed profile feature comes from the flower recognition of human, such as number of petals, colors and edges of a flower. Our best result is 83.57% in accuracy, which is obtained by aggregating several classification models with inception-v3 model. The size of the profile feature is 26 words (1 word = 32 bits), and the other features in SVM have size thousands of words. Note that a full image (200×200) with size thirty thousands words is used in inception-v3 model. Thus, the proposed profile feature is very effective in storage size for improving the classification accuracy.

***Index Terms*—** Flower Classification, Segmentation, Feature Extraction, Profile Feature, SVM, Adaboost, Inception-v3 Model.

1. INTRODUCTION

In our daily life, there is a variety of flowers everywhere, but it is a pity that sometimes we cannot tell what kinds of flowers they are. To recognize these flowers, one may go through many books and conduct some survey. It may be time-consuming for recognizing flowers manually. However, the story may be different if the task can be solved by computers or mobile devices automatically. One no longer needs to know all kinds of flowers; instead, a computer can be trained to do the same work.

In this paper, two kinds of features, color and texture, are used to perform the flower classification work. Color features consist of *HSV* (hue, saturation and value) [1, 2] histogram and *CIELAB* [3] histogram; texture features contain *SIFT* (scale-invariant feature transform) [4, 5], *SURF* (speeded up robust features) [6, 7] and our proposed features, named "Profile". The profile features include the ratio of the longest length and shortest length of a flower, number of petals, color set of a flower center and edge of a flower and moment feature. We use various combinations of the above features for training a good flower classification model. Finally, we achieve 83.57 % accuracy on Oxford-102 flowers dataset [8].

This paper is organized as follows. We introduce the relevant research of image classification in Section 2, and our method is presented in Section 3. Our experiments are shown in Section 4. Finally, our conclusion is given in Section 5.

2. PRELIMINARIES

Nilsback and Zisserman [9] proposed a flower classification method in 2008. They used some various combinations of features to improve the classification accuracy and obtained a good performance. Before extracting features, they divided each flower image into flower and background regions. Then they extracted the HSV, *HOG* (histogram of oriented gradients) [10] and SIFT descriptors. Finally, they trained a classification model with SVM [11, 12] and got the classification accuracy 72.8% on Oxford-102 flowers dataset. In 2009, Nilsback [13] added *CLAY* (color layout) feature and *HLAY* (HOG layout) feature to improve their method. He trained a classification model like the previous method and finally achieved 76.3% accuracy on Oxford-102 flowers dataset.

In 2010, Ito *et al.* [14] proposed a method based on HOG and co-occurrence features. The co-occurrence feature is usually more effective than the single feature because the co-occurrence feature of two events provides more information. For example, a color histogram shows the frequency of each color only, while the co-occurrence color histogram provides the frequency of co-occurrence of pairs of colors. They proposed three features based on co-occurrence feature, *color-CoHoG* (color-co-occurrence histogram of oriented gradients), *CoHED* (co-occurrence between edge orientation and color difference) and *CoHD* (co-occurrence histogram of pairs of edge orientations and color difference). Then they trained a *LIBLINEAR* [15] model with these three features and achieved 74.80% accuracy on Oxford-102 flowers dataset.

In 2011, Chai *et al.* [16] proposed a co-segmentation method, *BiCoS* (bi-level co-segmentation). Segmentation is an important step in image classification. Segmentation focuses on the object and improves the accuracy by removing the noise in the image. They used the GrabCut [17, 18, 19] method to segment an image into foreground and background regions. Every superpixel is assigned a label with either foreground or background. They collected all superpixel labels and trained a linear SVM to recognize the attribute of a region, either foreground or background. They generated the *BOW* (bag-of-words) histogram of *Lab* color and SIFT features. They trained an SVM model with the two features and achieved 80.00% accuracy on Oxford-102 flowers dataset.

In 2013, Angelova *et al.* [20] proposed an efficient object detection and segmentation method. They used *RGB* (red, green and blue) and HOG as features to train the SVM model for dividing an image into foreground and background regions. Then they used the same features from foreground regions to train the SVM classifier and finally achieved 80.66% accuracy on Oxford-102 flowers dataset.

Later in the recent years, deep learning is more mature and often gives people a surprising result. Contrast to traditional methods, deep learning focuses on the architecture of neural network and the computation. In 2015, Yoo *et al.* [21] proposed a multi-scale pyramid pooling method for better utilization of neural activations from a pre-trained *CNN* (convolution neural network). They achieved 91.28% accuracy on Oxford-102 flowers dataset.

In 2016, Liu *et al.* [22] trained a CNN model to solve the problem of flower classification. The architecture of their CNN model contains eight layers with weights. The first five layers are convolutional layers and the rest layers are fully-connected layers. They used local response normalization, overlapping pooling and dropout method to improve the accuracy of flower classification. They achieved 84.02% accuracy on Oxford-102 flowers dataset.

In 2017, Xia *et al.* [23] used the transfer learning to train the *inception-v3* model [24, 25] to classify the flowers. They achieved 94.00% accuracy on Oxford-102 flowers dataset.

Though the recent methods based on deep-learning or TensorFlow for flower classification may achieve an excellent accuracy, these methods need more data and time for training their models, and may not be so practical in some cases.

3. THE PROPOSED ALGORITHM

In this section, we present our method in which the profile feature is added. The size of the profile feature is 26 words only, but it increases the flower classification accuracy about 2% in the SVM model and about 3% in the ensemble model. The improvement is effective with such little increment in feature size. The method is divided into three stages, including image segmentation, feature extraction and classification model training.

3.1. Image Segmentation

Before extracting the features, the segmentation process is to partition each image into the foreground and background regions, and then remove the background region. We utilize the GrabCut [17] algorithm in OpenCV [26] to extract the foreground region. The GrabCut [17] algorithm needs a boundary rectangle that encloses the foreground. The detailed process is presented as follows.

Step 1: We set the initial rectangle as the whole image. That is, its four boundary lines are placed on the top, left, bottom and right of the image. For each boundary, the similarity from the current line to its next inner line by means of RGB colors is calculated. If the similarity is less than a threshold, it means that there is significant difference and the search process stops for this boundary. On the other hand, the search process goes on by shifting the boundary line toward the inner part. After the process has been carried out on the four boundary lines, a shrunk rectangle that covers the foreground is obtained.

Step 2: We build the second rectangle by extension starting from the center point. We use Laplacian zero crossing method to locate the boundaries in the image. For each direction, we find the most appropriate candidates. We then get a rectangle by combining these points.

Step 3: We average the two rectangles obtained in Steps 1 and 2 to get a refined rectangle for representing the boundary of the foreground region.

Step 4: The GrabCut program in OpenCV is applied to extract the foreground region with the boundary rectangle.

Figure 1 shows an example for illustrating the segmentation process.

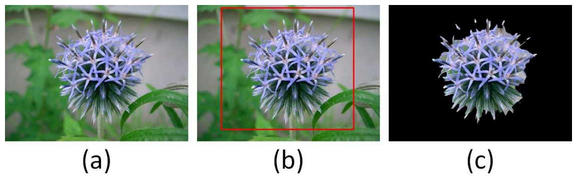
Fig. 1. The image segmentation process. (a) An original image. (b) The boundary of the foreground region. (c) The result of segmentation.

Table 1. The feature sets used in this paper and their sizes (in words). The various sizes of a feature set are obtained with different scales. 1 word denotes 32 bits.

Feature name	Size 1	Size 2	Size 3	Size 4
HSV	692	346	180	90
CIELAB	768	384	192	
SIFT	400			
SURF	400			
Profile	26			
Whole image $(200 \times 200 \times 3 \div 4)$				30000 words

3.2. Feature Extraction

After an image is segmented, we extract the features from the image. The features used in this paper are listed in Table 1.

For feature representation, we use the histogram to denote the HSV feature and CIELAB feature. The values in HSV are H, S, and V, where four different scales are used. For the first size of HSV, the range of H is between 0 and 179; the range of S is between 0 and 255; the range of V is between 0 and 255. Thus, the first size, 692, is calculated with $180 + 256 + 256$. The second size, 346, is calculated with $90 + 128 + 128$. The third size, 180, contains only the H value. The last size, 90, uses the H value only, whose dimension is reduced from 180 to 90.

Compared with the RGB color model, the CIELAB one is closer to our eye perception. Connolly and Fleiss [3] proposed an easy way to transform RGB values to CIELAB values. The values in the CIELAB color space are given as follows [3].

- L represents the luminance.
- a represents the color between green and red.
- b represents the color between blue and yellow.

For the first size of CIELAB, the range of the three values is between 0 and 255. Thus, the first size, 768, is calculated with $256 + 256 + 256$. The second size, 384, is calculated with $128 + 128 + 128$, whose values are reduced from 256 to 128. The last size, 192, is calculated with $64 + 64 + 64$, whose values are reduced from 256 to 64.

Lowe proposed the *scale-invariant feature transform* (SIFT) for image recognition, which has great effect for object recognition, image stitching, 3D modeling, etc. [4, 5]. It consists of four main stages, including scale-space extrema detection, key point localization, orientation assignment, and key point descriptor.

Speeded up robust features (SURF) [6] is a local feature detector and descriptor. It is claimed that SURF was designed to improve SIFT, so SURF is similar to SIFT in several ways. The algorithm is composed of four stages, including image integration, fast Hessian detection, orientation assignment and key point descriptor.

We use the *bag-of-words* [5] histogram to represent SIFT and SURF features. According to some primitive experiments, we set the dictionary size as 400 for both SIFT and SURF features. Inception-v3 model is an CNN model with a scaled image as its input. We use the transfer learning to transform the last layer to the Oxford-102 flower dataset. The original image size is about 400×600 and the scaled image size is 200×200. So, the input size of inception-v3 is about $200 \times 200 \times 3 \div 4 = 30000$ words (RGB and 4 bytes per word). In Table 1, the unit of size is "word", where one "word" is represented by 4 bytes.

3.3. Profile Feature

We propose the *profile* feature to describe the entire flower in the global view. We extract the contour of a flower image as follows.

Step 1: After segmentation and removing background, If there are more than one distinct object in an image, we find the biggest one as the target. Then we calculate the center point of the target.

Step 2: We calculate the distance from the center point to every contour point from degree 0 to degree 359. Then we get 360 contour distances.

Step 3: Find the first three longest distances l_1, l_2 and l_3 among the 360 contour distances. And find first three shortest distances s_1, s_2 and s_3. Then calculate three length ratios by $\frac{s_1+s_2+s_3}{3l_1}$, $\frac{s_1+s_2+s_3}{3l_2}$ and $\frac{s_1+s_2+s_3}{3l_2}$.

Figure 2 shows the steps of the above contour feature extraction. We extract some features from the contour and original image as the profile feature in the following. Table 2 shows the detailed size of each feature in the profile feature set.

1. Color: The center point and contour have different colors for some flowers, such as sunflower and pink primrose. We use HSV and CIELAB (six values) to represent the colors of center point and contour points. These values

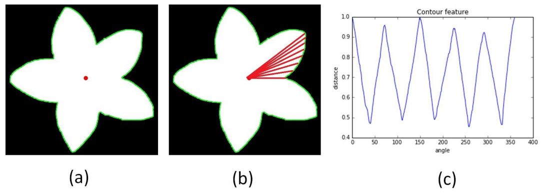

Fig. 2. The extraction of the 360-dimensional contour feature. (a.) The center point marked by the red color. (b) The distance from each contour point to the center illustrated by the red lines. (c.) The time series data constructed from the 360 normalized contour distances.

Table 2. The size of the profile feature.

Feature name	Size (word)
Color	12
Length	3
Hu's moment	7
Similarity	3
Petals	1
Total	26

of the center point are got by averaging 3×3 subimage in the center. These values of the contour points are obtained by averaging all contour points with thickness 3 pixels.

2. Length: The three length ratios are got from the above contour feature.

3. Hu's moment [27]: We use the Hu's seven moments to describe the shape of flower. The moments are normalized with respect to the geometric moments and are arranged with respect to its center, so the moments have the attributes of scale and rotation invariance.

4. Similarity between the flower shape and a circle: Some flowers are circular in shapes but some are not. We extract the longest, shortest and average contour distances. Next, use these three distances as radiuses to calculate three circle areas. Then compute the three ratios from the three circle areas and the flower area. These three area ratios are viewed as the shape similarities heuristically.

5. Number of petals: We get the number of petals according to the contour of flower.

3.4. Classification Models

After finishing the features extraction, the subsequent work is to train classification models, SVM [11, 12] and Adaboost [28, 29, 30, 31, 32, 33, 34, 35], with these features. We then classify all testing flower images and calculate the classification accuracy.

Among all SVM software tools, LIBSVM [12] is widely used in academic researches. In this paper, we invoke LIBSVM to build SVM models. We try several parameter combinations for the SVM classifiers. Table 3 shows the range of the SVM parameters.

Table 3. The SVM parameters used in this paper.

Parameter	Value
SVM-type	C-SVC
kernel	Radial basis function
cost	1 to 100
γ	0.01 to 1

Boosting [35] is a method for creating a highly accurate predication model by combining several weak and inaccurate learning methods. In 1996, Freund and Schapire [28] proposed the *Adaboost* [28, 29, 30, 31] method. Adaboost is an abbreviation of "Adaptive Boosting", which means that the error yielded from the previous classifier can be used to train the next classifier. Table 4 shows the parameters of Adaboost.

Table 4. The parameters of Adaboost used in this paper.

Parameter	Value
classifier	Random forest
numiterations	10 to 20
weightThreshold	50 to 100

4. EXPERIMENTAL RESULTS

In this section, we show our experimental results. These experiments were performed on a PC equipped with an Intel(R) Core(TM) i5-2400 CPU 3.10GHz and 20G RAM. Our experimental dataset is the Oxford-102 flower dataset, containing totally 8189 labeled images in 102 categories, in which each category has 40 to 258 images. Figure 3 lists the 102-category flowers.

To evaluate the classification accuracy, we divide the dataset into three datasets, the training, validation and testing datasets. There are 10 images in each category for training, 10 images in each category for validation and the rest of the images for testing. We use the validation set to tune the parameters in our experiments, as shown in Table 3 and Table 4. Then the above process is repeated for 100 times.

The *accuracy* is defined as the ratio of correct classifications over the total number of classifications. And, in the following tables, each classification accuracy is the average of 100 experimental results. Table 5 shows the accuracies of SVM, Adaboost and inception-v3 with various features.

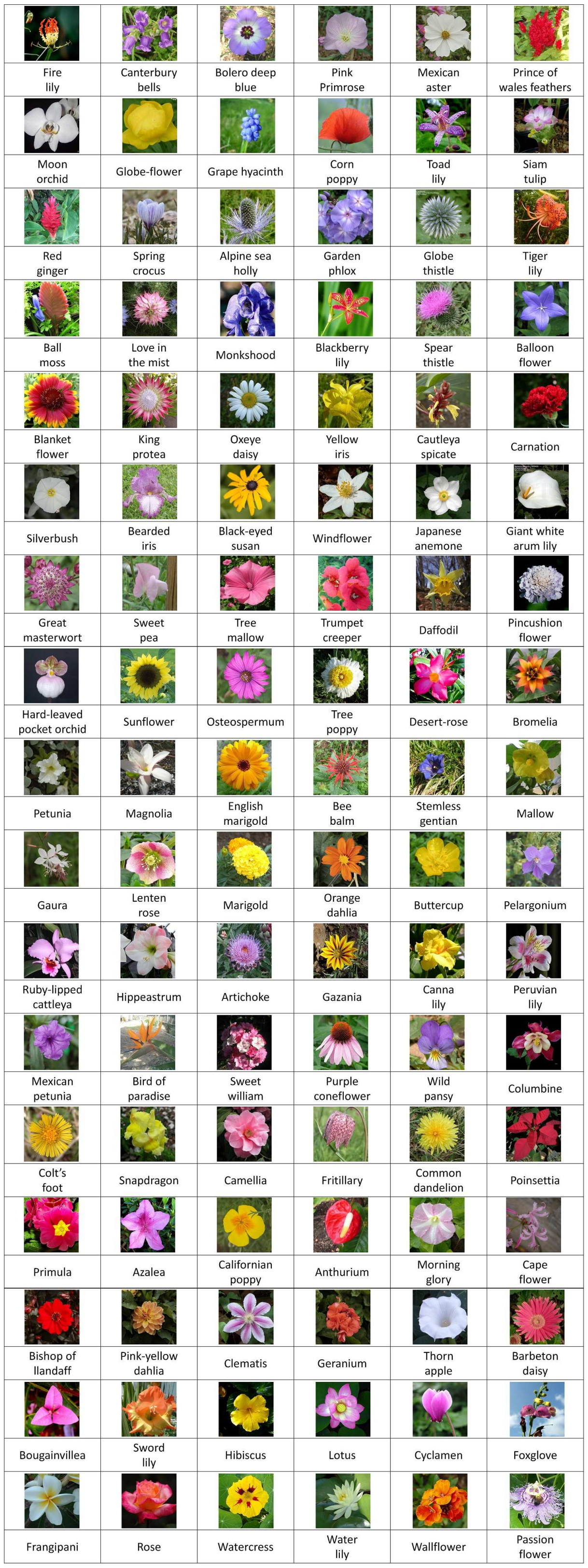

Fig. 3. The Oxford-102 flower dataset.

Table 5. The accuracies of SVM, Adaboost and inception-v3 with various features.

Feature name	Size (word)	Accuracy		
		SVM	Adaboost	Inception-v3
CIELAB	384	29.27%	43.25%	-
HSV	90	21.43%	33.16%	-
SIFT	400	49.04%	28.51%	-
SURF	400	54.91%	35.72%	-
Whole image	30000	-	-	77.48%
Profile	26	30.25%	34.59%	-

Compared with other features, the size of the proposed profile feature is tiny. The accuracy of the profile feature is comparable to other features, except Inception-v3, with larger size.

Table 6 shows the accuracies of various feature sizes classified by SVM. Table 7 shows accuracies of various feature sizes classified by SVM and Adaboost with the weighted vote. The item inside parenthesis denotes the feature name, where in front of the parenthesis is classifier name, and the number following the feature is its size (dimension). For example, "Adaboost(CIELAB384)" means that we use CIELAB with size 384 to train an Adaboost model.

In Table 6, "with profile" means that the profile feature and other features are combined together as an integrated feature within an SVM classifier, represented as the "&" operator. In Table 7, "with profile" means that the profile feature is used in an individual SVM, and the classification results are obtained by the weighted vote manner from several classifiers, represented as the "+" operator. That is, each flower class j of a classifier i is assigned a class weight w_{ij}, which is given by the classification accuracy during the training process. In Tables 6 and 7, " with inception-v3" is combined by the "+" operator with other classifiers.

During the classification stage, for a testing image I, a classifier i exports a class label j as well as its associated probability p_{ij}. Then, the classified label of I is obtained by the following weighted vote.

$$\text{classified label of } I = \arg\max_j \left(\sum_i w_{ij} \times p_{ij}\right). \qquad (1)$$

In Table 6, the SVM model without inception-v3 model can be improved about 1.8% accuracy if we add the profile feature. Furthermore, the SVM model with inception-v3 model can be improved about 0.8% accuracy with the profile feature. In Table 7, our ensemble method without inception-v3 model can be improved about 2.9% accuracy with the profile feature. Additionally, our ensemble method with the profile feature and inception-v3 model can be improved about 0.5% accuracy. Table 8 shows the highest ac-

Table 6. The accuracies of various feature sizes classified by SVM. Here, "&" means feature combination.

No.	Classifiers	Accuracy			
		without incpetion-v3		with inception-v3	
		without profile	with profile	without profile	with profile
1	SVM(CIELAB768& HSV692&SIFT400& SURF400)	67.21%	69.00%	81.34%	82.18%
2	SVM(CIELAB768& HSV346&SIFT400& SURF400)	67.48%	69.26%	81.46%	82.29%
3	SVM(CIELAB768& HSV180&SIFT400& SURF400)	67.52%	69.44%	81.58%	82.32%
4	SVM(CIELAB768& HSV90&SIFT400& SURF400)	67.60%	69.57%	81.67%	82.46%
5	SVM(CIELAB384& HSV692&SIFT400& SURF400)	68.39%	69.83%	81.81%	82.56%
6	SVM(CIELAB384& HSV346&SIFT400& SURF400)	68.42%	70.12%	82.03%	82.67%
7	SVM(CIELAB384& HSV180&SIFT400& SURF400)	68.45%	70.19%	82.19%	82.79%
8	SVM(CIELAB384& HSV90&SIFT400& SURF400)	68.56%	70.33%	82.34%	83.01%
9	SVM(CIELAB192& HSV692&SIFT400& SURF400)	67.37%	69.23%	81.49%	82.24%
10	SVM(CIELAB192& HSV346&SIFT400& SURF400)	67.48%	69.37%	81.57%	82.43%
11	SVM(CIELAB192& HSV180&SIFT400& SURF400)	67.52%	69.46%	81.63%	82.52%
12	SVM(CIELAB192& HSV90&SIFT400& SURF400)	67.65%	69.64%	81.77%	82.63%

curacy of the ensemble combined from several classifiers and other researches.

The profile feature is effective and compatible to other feature with larger size, such as HSV or CIELAB. We think that the profile feature is useful in some aspects. First, the colors in the profile feature are practical, since many flowers have different colors in the center and the contour. We use this characteristic to classify these flowers and return good results. Second, the number of petals in a flower is an obvious feature since different types of flowers usually have different numbers of petals.

5. CONCLUSIONS AND FUTURE WORK

The inception-v3 model proposed by Xia *et al.* [23] can achieve a very high accuracy of 94%. However, in our implementation, we cannot achieve such high accuracy. We get only 77.48%, as shown in Table 5. We think that more parameters in inception-v3 need to be tuned.

In this paper, we achieve the acceptable accuracy of 83.57% and outperforms all methods without deep-learning. We propose an effective feature, named the "Profile" feature. The "Profile" feature subset itself is not very prominent, but it contains some complement information and thus may improve classification accuracy when combining with other features.

In the future, we may improve our method in three possible ways. First, we may segment other profile features such

Table 7. The accuracies of various feature sizes classified by SVM and Adaboost. Here, "+" means the combination of classification results with the weighted vote.

No.	Classifiers	Accuracy			
		without incpetion-v3		with inception-v3	
		without profile	with profile	without profile	with profile
1	Adaboost(CIELAB768)+ Adaboost(HSV692)+ SVM(SIFT400)+ SVM(SURF400)	53.26%	56.41%	82.31%	82.89%
2	Adaboost(CIELAB768)+ Adaboost(HSV346)+ SVM(SIFT400)+ SVM(SURF400)	53.47%	56.54%	82.39%	82.94%
3	Adaboost(CIELAB768)+ Adaboost(HSV180)+ SVM(SIFT400)+ SVM(SURF400)	53.51%	56.73%	82.47%	83.00%
4	Adaboost(CIELAB768)+ Adaboost(HSV90)+ SVM(SIFT400)+ SVM(SURF400)	53.66%	56.92%	82.53%	83.07%
5	Adaboost(CIELAB384)+ Adaboost(HSV692)+ SVM(SIFT400)+ SVM(SURF400)	54.01%	56.86%	82.76%	83.21%
6	Adaboost(CIELAB384)+ Adaboost(HSV346)+ SVM(SIFT400)+ SVM(SURF400)	54.23%	56.95%	82.88%	83.34%
7	Adaboost(CIELAB384)+ Adaboost(HSV180)+ SVM(SIFT400)+ SVM(SURF400)	54.39%	57.14%	82.94%	83.39%
8	Adaboost(CIELAB384)+ Adaboost(HSV90)+ SVM(SIFT400)+ SVM(SURF400)	54.58%	57.33%	83.09%	83.57%
9	Adaboost(CIELAB192)+ Adaboost(HSV692)+ SVM(SIFT400)+ SVM(SURF400)	53.60%	56.52%	82.43%	82.95%
10	Adaboost(CIELAB192)+ Adaboost(HSV346)+ SVM(SIFT400)+ SVM(SURF400)	53.72%	56.58%	82.51%	83.04%
11	Adaboost(CIELAB192)+ Adaboost(HSV180)+ SVM(SIFT400)+ SVM(SURF400)	53.84%	56.67%	82.75%	83.25%
12	Adaboost(CIELAB192)+ Adaboost(HSV90)+ SVM(SIFT400)+ SVM(SURF400)	53.76%	56.65%	82.61%	83.18%

Table 8. The accuracies of various classifier combinations.

Method	Accuracy				Note
	without inception-v3		with inception-v3		
	without profile	with profile	without profile	with profile	
SVM(CIELAB384& HSV90& SIFT400& SURF400)	68.56%	70.33%	82.34%	83.01%	-
AdaBoost(CIELAB384)+ AdaBoost(HSV90)+ SVM(SIFT400)+ SVM(SURF400)	54.58%	57.33%	83.09%	83.57%	-
Nilsback and Zisserman 2008	72.80%				HOG, HSV, SIFT, SVM
Nilsback 2009	76.30%				CLAY, HLAY, HOG, HSV, SIFT, SVM
Ito and Kubota 2010	74.80%				CoHED, CoHD, Color-CoHOG, Color histogram, LIBLINEAR
Cha *et al.* 2011	80.00%				BiCoS-MT, Lab color, SIFT, SVM
Angelova *et al.* 2013	80.66%				HOG, Liblinear SVM, LLC
Yoo *et al.* 2015	91.28%				CNN, Fisher kernel
Liu *et al.* 2016	84.02%				CNN, Luminance map Saliency map
Xia *et al.* 2017	94.00%				Tensorflow, Inception-v3

as petals, calyxes or stamens. Then use shape or color to classify what the flower is. Second, we may adjust our ensemble model by replacing the voting method. We may try the *behavior knowledge space* (BKS) to classify the flowers. Third, we may put the extracted features of flowers into the neural network for classification, instead of the whole flower image.

REFERENCES

[1] Raphael Gonzalez and Richard E. Woods, *Digital Image Processing*, Prentice Hall Press, 2002.

[2] Mark D Fairchild, *Color Appearance Models*, Addison-Wesley, Boston, USA, 2005.

[3] C Connolly and T. Fleiss, "A study of efficiency and accuracy in the transformation from RGB to CIELAB color space," *IEEE Transactions on Image Processing*, vol. 6, no. 7, pp. 1046–1048, July 1997.

[4] David G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the 7th IEEE International Conference on Computer Vision*, Kcrkyra, Greece, 1999, vol. 2, pp. 1150–1157.

[5] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[6] Herbert Bay, Andreas Ess, and Tinne Tuytelaars, "SURF: Speeded up robust features," *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, 2008.

[7] D. Agnew, "Efficient use of the Hessian matrix for circuit optimization," *IEEE Transactions on Circuits and Systems*, vol. 25, pp. 600–608, 1978.

[8] Maria Elena Nilsback and Andrew Zisserman, "102 category flower dataset," 2008, http://www.robots.ox.ac.uk/ vgg/data/flowers/102/.

[9] Maria Elena Nilsback and Andrew Zisserman, "Automated flower classification over a large number of classes," in *Proceedings of the 6th Indian Conference on Computer Vision, Graphics and Image Processing*, Bhubaneswar, India, 2008, pp. 722–729.

[10] Navneet Dalal and Bill Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, 2005, vol. 1 of *CVPR '05*, pp. 886–893, IEEE Computer Society.

[11] Corinna Cortes and Vladimir Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sept. 1995.

[12] Chih Chung Chang and Chih Jen Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[13] Maria Elena Nilsback, *An Automatic Visual Flora: Segmentation and Classification of Flower Images*, Ph.D. thesis, University of Oxford, Oxford, England, UK, 2009.

[14] Satoshi Ito and Susumu Kubota, "Object classification using heterogeneous co-occurrence features," in *Proceedings of the 11th European Conference on Computer Vision: Part II*, Berlin, Heidelberg, 2010, ECCV'10, pp. 209–222, Springer-Verlag.

[15] Rong En Fan, Kai Wei Chang, Cho Jui Hsieh, Xiang Rui Wang, and Chih Jen Lin, "LIBLINEAR: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, June 2008.

[16] Yuning Chai, V. Lempitsky, and A. Zisserman, "BiCoS: A bi-level co-segmentation method for image classification," in *Proceedings of 2011 IEEE International Conference on Computer Vision*, Barcelona, Spain, Nov. 2011, pp. 2579–2586.

[17] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake, "Grabcut - interactive foreground extraction using iterated graph cuts," in *Proceedings of ACM SIGGRAPH 2004*, Los Angeles, California, USA, Aug. 2004, vol. 23, pp. 309–314.

[18] Yuri Boykov, Olga Veksler, and Ramin Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001.

[19] Yuri Boykov and Garet Funka-Lea, "Graph cuts and efficient N-D image segmentation," *International Journal of Computer Vision*, vol. 70, no. 2, pp. 109–131, Nov. 2006.

[20] Anelia Angelova, Shenghuo Zhu, and Yuanqing Lin, "Image segmentation for large-scale subcategory flower recognition," in *Proceedings of 2013 IEEE Workshop on Applications of Computer Vision*, Tampa, FL, USA, 2013, pp. 39–45.

[21] D. Yoo, S. Park, J. Y. Lee, and In So Kweon, "Multi-scale pyramid pooling for deep convolutional representation," in *Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Boston, MA, USA, June 2015, pp. 71–80.

[22] Y. Liu, F. Tang, D. Zhou, Y. Meng, and W. Dong, "Flower classification via convolutional neural network," in *Proceedings of 2016 IEEE International Con-*

ference on Functional-Structural Plant Growth Modeling, Simulation, Visualization and Applications, Qingdao, China, Nov. 2016, pp. 110–116.

[23] Xiaoling Xia, Cui Xu, and Bing Nan, "Inception-v3 for flower classification," in *Proceedings of 2017 2nd IEEE International Conference on Image, Vision and Computing*, Chengdu, China, 2017, pp. 783–787.

[24] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, 2016, pp. 2818–2826.

[25] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, Savannah, GA, USA, 2016, pp. 265–283, USENIX Association.

[26] Itseez, "Open source computer vision library," 2015, https://github.com/itseez/opencv.

[27] Ming Kuei Hu, "Visual pattern recognition by moment invariants, computer methods in image analysis," *IRE Transactions on Information Theory*, vol. 8, 1962.

[28] Yoav Freund and Robert E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of 13th International Conference on Machine Learning*, San Francisco, USA, 1996, pp. 148–156, Morgan Kaufmann.

[29] Yoav Freund and Robert E Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[30] Robert E. Schapire, "A brief introduction to boosting," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, San Francisco, CA, USA, 1999, vol. 2 of *IJCAI'99*, pp. 1401–1406, Morgan Kaufmann Publishers Inc.

[31] Robert E. Schapire, *Explaining AdaBoost*, pp. 37–52, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[32] Paul E. Utgoff, "Incremental induction of decision trees," *Machine Learning*, vol. 4, no. 2, pp. 161–186, Nov. 1989.

[33] Tin Kam Ho, "Random decision forests," in *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, Washington, DC, USA, 1995, vol. 1 of *ICDAR '95*, pp. 278–282, IEEE Computer Society.

[34] Leo Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[35] M. Kearns, "Thoughts on hypothesis boosting," Machine Learning class project, Dec. 1988.