Flower Classification Based On Image Processing And Machine Learning Techniques

Satyam Keshri-203070010, Nipadkar Subham Sanjay-204090007, M. V. Mani Kumar-203070012
Project report for IITB EE610 Image Processing 2021
Department of Electrical Engineering, Department of Mathematics
Indian Institute of Technology, Bombay, India
203070010@iitb.ac.in, 204090007@iitb.ac.in, 203070012@iitb.ac.in

***Abstract*—Since there are plenty flowers, it is important to develop a model which can classify flower so that this classification will not require a specialist. In this project we developed a Machine Learning model for the classification of flowers. The features used for the Machine Learning model are extracted using image processing techniques like histogram for Red, Green, Blue, Hue and Saturation characteristics, Canny Edge Detection, Hu's 7 moments. We are using Support Vector Machine for Classification, Random Forest Classification, Decision Tree Classification and K Nearest Neighbor Classification as Machine Learning models. Histogram is used to get colour characteristics. Hu's 7 moments algorithm is used to get the edge characteristics. We observed that Support Vector Machine for multi-class classification has f1 score $= 0.71875$. Random Forest Classification for multi-class classification, Decision Tree Classification for multi-class classification and K Nearest Neighbor Classification for multi-class classification has accuracy score as 0.734375, 0.578125 and 0.5625 respectively. Thus, the best model will be Random Forest Classification for multi-class classification with number of trees $= 200$ and tree depth $= 10$.**

I. Introduction

There are many flower species on the planet. A few varieties have many colours, like roses. It is difficult to recollect all flower names and their data. Besides, somebody might be mistaken for comparative flower species. For instance, Red rose and white rose have comparative names and petal shapes however they have various tones and petal lengths.

As of now it is exceptionally difficult to recognize specific flowers or flower species in some other manner. However, we can use our personal resources and experience of specialists. Accessibility of such specialists and resources some times might be difficult. Looking for such data on the Internet is, today, especially limited to keywords; text processing. Indeed, even in this the searcher needs to give adequately helpful catchphrases, which they can't do, which is the essence of the matter.

So in this project we developed a Machine Learning model that will help in the classification of flowers. The user have to give the image of flower as a input to the model. Model output is the name of the flower species.

In this project we have used classical image processing techniques to extract the features of the flowers and proper characteristics of RGB and HSV domains are also extracted using histogram normalization. The features extracted are used in training the Machine Learning model.

In the next section we will discuss background and prior work done regarding topic of project. Then in the next section we will describe our data and methodology used. We will also describe Machine Learning models used in the project. Further in the next section we will discuss in detail results obtained. Then further in the next section we will describe our conclusions, learning and what future work can be done. Our project is mainly based on [5]

II. Background and prior work

A. Colour based system

This method was proposed by Das et al[1].This algorithm classifies flowers based on colours.But the disadvantage of this method is many different species of flowers have same colour and same species of flowers have different colours.

B. Wild flower recognisation system

This method was proposed by Saitoh and Kaneto et al["Automatic recognition of wild flowers"33"Automatic recognition of wild flowers"]. In this method piece wise linear discriminant function is used and this method requires images of flower along with leaf.The features of leafs and flowers are used. This method require both time and space complexity.

C. Colour and Shape Based System

This method was proposed by Nilsback and Zisserman et al[3]. In this method graph cut along with colour dependent random forest algorithms is used. Shape fitting methods are used for extracting the petal characteristics. In this method overlapping petals and finding centre of flower is a disadvantage of this method.

D. Segmentation and Sub regions Based System

This method was proposed by Chai et al[4]. In this method segmentation and sub region division of input image along with discriminative classifier and grabcut algorithms were used. By proper feature selection and combinations of SVM and co-segmentation algorithms. This method is computationally expensive.

III. Data and Methodology

We have used Python 3.8.8 programming language and Jupyter Notebook 6.3.0 for coding.

A. Dataset of Flowers

The dataset which we used in this project consists of 320 images of flowers. These images correspond to total of 4 flower species. These 4 species are Daffodil, Snowdrop, Lily Valley and Bluebell. Corresponding to each species we have 80 images in the dataset. The features are extracted from these images. We made a dataset consisting of these features which is used for Machine Learning models. We will discuss about this further.

B. Feature Extraction

1) Conversion from BGR to RGB: Since we used $OpenCV - Python$ library to import images, all the images are in BGR format. We converted these images to RGB format.

2) Resizing: We resized all images in the dataset to half of their original size. We do this so that the processing time is reduced. These resized images are used in further image processing.

3) Converting to Grayscale: We converted all resized images into grayscale images. For this purpose, we used following formula:

$$Y = 0.2126 \times R + 0.7152 \times G + 0.0722 \times B$$

where, R denotes the value of pixel corresponding to Red channel, G denotes the value of pixel corresponding to Green channel, B denotes the value of pixel corresponding to Blue channel and Y denotes the value of pixel in grayscale image.

4) Characteristics Through Histograms: In this step we extract the characteristics regarding colour of the flower. We converted all resized images from RGB to HSV format. From resized RGB images, we extracted Red, Green and Blue channel values. From HSV images, we extracted Hue and Saturation channel values. Then we obtained histogram corresponding to these channels for each image. The histogram is obtained such that it has 8 bins for each channel. We normalized the obtained histogram. Thus, for each image we obtained 40 features. These obtained features are later utilized in the Machine Learning models.

5) Canny Edge Detection: We applied this algorithm to get edge data about resized images. For applying this algorithm, we considered grayscale images. We applied Gaussian blur kernel of size 5×5 and used mirror padding. We blurred images so that if any noise is present in the images, it will be reduced. On these blurred images we used Sobel operator of size 3×3 and mirror padding to get gradient values along x direction and y direction. From these gradient values, we obtained magnitude and angle (in degrees). We set the weak threshold value as 10% of maximum magnitude and strong threshold value as 50% of maximum magnitude. For each pixel, we converted angles such that if the absolute value of angle was less than 180 degree, we considered angle as its absolute value. Otherwise, we considered angle as absolute value of difference between angle and 180 degree. Then we divided angle into partitions of 0 to 22.5, 22.5 to 67.5, 67.5 to 112.5, 112.5 to 157.5 and 157.5 to 180. Then we applied non-maximum suppression. If the magnitude for a pixel was less than the weak threshold value, we set it to 0. These output images are further used for feature extraction. The code for Canny Edge Detection is taken from [20]

6) Hu's seven moment: The images obtained after applying the Canny Edge Detection algorithm are used to obtain Hu's seven moments. If we had $m \times n$ image and pixel value at (x, y) is given as $f(x, y)$, then the $(p, q)^{th}$ raw moment of image is given as,

$$m_{p,q} = \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} x^p y^q f(x, y)$$

The $(p, q)^{th}$ central moment of image is given as,

$$\mu_{p,q} = \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} (x - \bar{x})^p (y - \bar{y})^q f(x, y)$$

where, $\bar{x} = \frac{m_{1,0}}{m_{0,0}}$ and $\bar{y} = \frac{m_{0,1}}{m_{0,0}}$. We considered, $\mu_{1,1}$, $\mu_{2,0}$, $\mu_{0,2}$, $\mu_{2,1}$, $\mu_{1,2}$, $\mu_{3,0}$ and $\mu_{0,3}$. These 7 moments can also be expressed as follow:

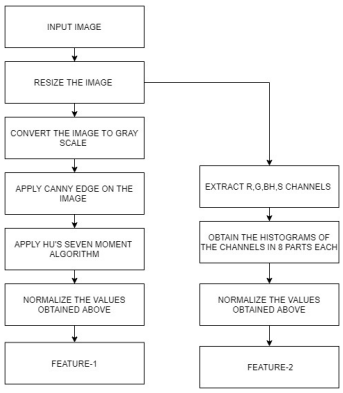
$$\begin{aligned}
\mu_{1,1} &= m_{1,1} - \bar{x} m_{0,1} \\
\mu_{2,0} &= m_{2,0} - \bar{x} m_{1,0} \\
\mu_{0,2} &= m_{0,2} - \bar{y} m_{0,1} \\
\mu_{2,1} &= m_{2,1} - 2\bar{x} m_{1,1} - \bar{y} m_{2,0} + 2\bar{x}^2 m_{0,1} \\
\mu_{1,2} &= m_{1,2} - 2\bar{y} m_{1,1} - \bar{x} m_{0,2} + 2\bar{y}^2 m_{1,0} \\
\mu_{3,0} &= m_{3,0} - 3\bar{x} m_{2,0} + 2\bar{x}^2 m_{1,0} \\
\mu_{0,3} &= m_{0,3} - 3\bar{y} m_{0,2} + 2\bar{y}^2 m_{0,1}
\end{aligned}$$

Then we applied scaling to these 7 moments and obtained $\eta_{p,q}$, where

$$\eta_{p,q} = \frac{\mu_{p,q}}{\mu_{0,0}^{\gamma}}$$

where, $\gamma = (p + q)/2 + 1$.
We obtained seven hu moment values of every image. Then we normalized these moments. These Hu's moments of images are also used as feature for Machine Learning models.

C. Flow chart of Feature Extraction



D. Dataset For Machine Learning

Till now we have discussed about feature extraction. By utilizing the above methods of feature extraction we have obtained 47 values for every image. As we know earlier, there are 4 species of flowers. So, we want to develop a model such that given input as features it will classify into 1 of the 4 species. We divided our dataset of features into two parts, $viz.$ training and testing, in such a way that 80% of data corresponding to each species is included in training part, while remaining 20% is included in testing part. Then, we divided training part into training and validation data, such that 80% of training part is included in training data and remaining 20% is included in validation data. Thus, we have balanced data, which means, we have equal number of observations corresponding to each class. This data was used for training, validation and testing of Machine Learning Models

E. Machine Learning Models

In this section we will discuss about various Machine Learning Models which we have implemented. We have implemented 4 Machine Learning Classification models $viz.$ Support Vector Machine for multi-class classification, Random Forest Classification for multi-class classification, Decision Tree Classification for multi-class classification and K Nearest Neighbor Classification for multi-class classification.

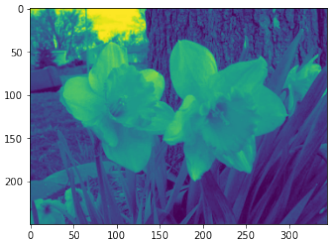
For Support Vector Machine for multi-class classification, we considered polynomial kernel. We considered hyper-parameters as C and $degree$. C is the regularization parameter and $degree$ is the degree of polynomial kernel. For Random Forest Classification for multi-class classification, we considered hyper-parameters as number of trees and tree depth. For Decision Tree Classification for multi-class classification, we considered criterion as $'entropy'$ and hyper-parameter as tree depth. For K Nearest Neighbor Classification for multi-class classification we considered distance metric as $'minkowski'$ and power parameter as 2. We considered hyper-parameter as number of neighbors.

To compare the performance of Machine Learning models, we will compute f1 score for Support Vector Machine for multi-class classification and accuracy score for other 3 models. While computing f1 score, we considered average as $'micro'$ because we have 4 classes for classification problem. Higher these scores are, better are the models.

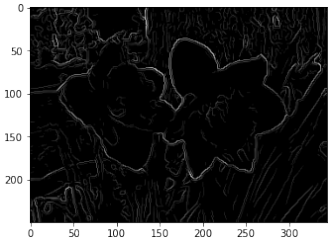
IV. Experiments And Results

The first image of the dataset of images in RGB format is displayed as follow:

The grayscale image corresponding to the above displayed image is as follow:



The Canny Edge image corresponding to the above displayed grayscale image is as follow:



The 7 Hu's moments obtained corresponding to first image are as follows:

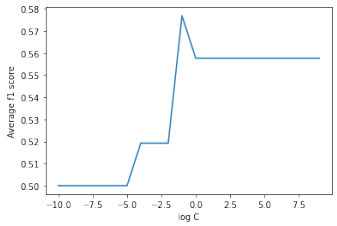
-5.01304143 $\times 10^{-4}$
5.68250880 $\times 10^{-3}$
9.99294553 $\times 10^{-3}$
-5.73638671 $\times 10^{-5}$
-4.95719336 $\times 10^{-5}$
1.38339481 $\times 10^{-4}$
1.72029440 $\times 10^{-4}$

The normalized 7 Hu's moments obtained corresponding to first image are as follows:

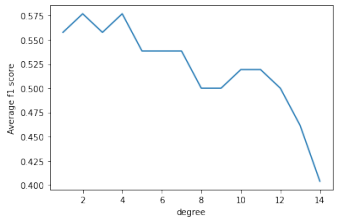
0
0.58925727
1
0.04230319
0.04304569
0.06095182
0.06416215

For each Machine Learning model, we considered some values of hyper-parameters. We fitted each model 20 times by considering these values of hyper-parameters on training data and obtained score for validation data. Then we took average of scores over 20 trial for each considered value of hyper-parameter. Then we plot average scores, so as to detect under-fitting and over-fitting of model. In this way, we obtained the best value for hyper-parameters.

1) Support Vector Machine for multi-class classification: While using Support Vector Machine for multi-class classification, we considered polynomial kernel. At first, we considered hyper-parameter as C. We considered the values for $log\,(C)$ as integers from -10 to 9. We fitted this model on training data for 20 trials and obtained f1 score corresponding to each value of hyper-parameter considered and each trial. Then we computed average of these f1 scores over trials. The graph of these average f1 scores for various values of hyper-parameter considered is as follows:

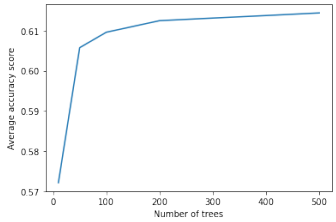


We can see that, before $log\,(C) = 0$, average f1 score has increasing trend. This indicates that before $log\,(C) = 0$ there is under-fitting. After $log\,(C) = 0$, average f1 score does not change significantly. Thus, the best value of $log\,(C)$ is 0. Thus, the best value of hyper-parameter C is 1.
Next we considered hyper-parameter as $degree$ with values as integers from 1 to 14. We fitted this model with $C = 1$ on training data for 20 trials and obtained f1 score corresponding to each value of hyper-parameter considered and each trial. Then we computed average of these f1 score over trials. The graph of these average f1 score for various values of hyper-parameter considered is as follows:

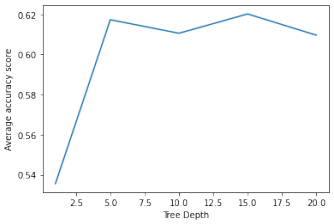


We can see that, after $degree = 3$ of polynomial kernel, average f1 score has decreasing trend. This indicates that after $degree = 3$, there is over-fitting. Thus, the best value of hyper-parameter $degree$ is 3.
Then we fitted the model with hyper-parameter $C = 1$ and $degree = 3$ on training data. We obtained f1 score for test data as f1 score $= 0.71875$.

2) Random Forest Classification for multi-class classification: While using Random Forest Classification for multi-class classification, at first we considered hyper-parameter as number of trees. We considered the values for number of trees as 10, 50, 100, 200 and 500. We fitted this model on training data for 20 trials and obtained accuracy score corresponding to each value of hyper-parameter considered and each trial. Then we computed average of these accuracy scores over trials. The graph of these average accuracy scores for various values of hyper-parameter considered is as follows:

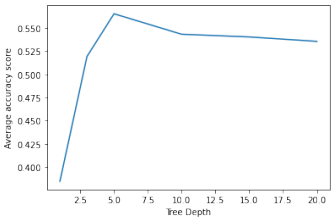


We can see that, before number of trees, average accuracy score has increasing trend. This indicates that before number of trees $= 200$, there is under-fitting. After number of trees $= 200$, there is no significant change in average accuracy score. Thus, the best choice for hyper-parameter number of trees is 200.
Next we considered hyper-parameter as tree depth with values as 1, 5, 10, 15 and 20. We fitted this model with number of trees $= 200$ on training data for 20 trials and obtained accuracy score corresponding to each value of hyper-parameter considered and each trial. Then we computed average of these accuracy scores over trials. The graph of these average accuracy scores for various values of hyper-parameter considered is as follows:



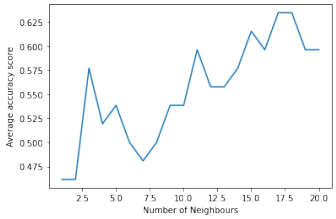
We can see that, before tree depth, average accuracy score has increasing trend. This indicates that before tree depth $= 10$, there is under-fitting. After tree depth $= 10$, average accuracy score does not change significantly. Thus, best choice for hyper-parameter tree depth is 10.
Then we fitted the model with hyper-parameter number of trees $= 200$ and tree depth $= 10$ on training data. We obtained accuracy score for test data as accuracy score $= 0.734375$.

3) Decision Tree Classification for multi-class classification: For Decision Tree Classification for multi-class classification, we considered hyper-parameter as tree depth. We considered the values for tree depth as 1, 3, 5, 10, 15 and 20. We fitted this model on training data for 20 trials and obtained accuracy score corresponding to each value of hyper-parameter considered and each trial. Then we computed average of these accuracy scores over trials. The graph of these average accuracy scores for various values of hyper-parameter considered is as follows:



We can see that, before tree depth $= 5$ average accuracy score has increasing trend. This indicates that before tree depth $= 5$, there is under-fitting. After tree depth $= 5$, average accuracy score has decreasing trend. This indicates that after tree depth $= 5$, there is over-fitting. Thus, the best choice of hyper-parameter tree depth is 5.
Then we fitted the model with hyper-parameter tree depth $= 5$ on training data. We obtained accuracy score for test data as accuracy score $= 0.578125$.

4) K Nearest Neighbor Classification for multi-class classification: For K Nearest Neighbor Classification for multi-class classification, we considered hyper-parameter as number of neighbors. We considered the values for number of neighbors as integers from 1 to 20. We fitted this model on training data for 20 trials and obtained accuracy score corresponding to each value of hyper-parameter considered and each trial. Then we computed average of these accuracy scores over trials. The graph of these average accuracy scores for various values of hyper-parameter considered is as follows:



We can see that, before number of neighbors $= 17$, average accuracy score has increasing trend. This indicates there is under-fitting before number of neighbors $= 17$. After number of neighbors $= 17$, average accuracy score has decreasing trend. This indicates there is over-fitting after number of neighbors $= 17$. Thus best choice of hyper-parameter number of neighbors is 17
Then we fitted the model with hyper-parameter number of neighbors $= 17$ on training data. We obtained accuracy score for test data as accuracy score $= 0.5625$.

V. Learning, Conclusions, and Future Work

From this project we were able to learn about image pre-processing techniques, feature extraction including Hu's 7 moments,implentation of canny edge and implementing Machine Learning Methods. We understood the concept of under-fitting and over-fitting of models. The f1 score or accuracy score corresponding to test data for the Machine Learning models considered is given as follow:

Sr. No.	Machine Learning Model	f1 score/Accuracy score
1	Support Vector Machine	71.87%
2	Random Forest	73.43%
3	Decision Tree	57.81%
4	K Nearest Neighbor	56.25%

Thus, we can see that Support Vector Machine and Random Forest Classification are giving f1 score and accuracy score respectively greater than 70%. The highest accuracy score is 73.43% corresponding to Random Forest Classification for multi-class classification. Thus, the best model is Random Forest Classification with number of trees $= 200$ and tree depth $= 10$.
In future if we include graph cut algorithms to make the background of images black, before proceeding to image processing techniques, then the accuracy of Machine Learning models may increase.

Contribution of Team Members

All three members read the research paper [5]. All three members equally contributed to carry out image pre-processing techniques like conversion from BGR to RGB, resizing of

images, converting to grayscale and conversion from RGB to HSV. Satyam Keshri implemented K Nearest Neighbor Classification for multi-class classification. Nipadkar Subham Sanjay implemented code for obtaining histogram with 8 bins for Red, Green, Blue, Hue and Saturation channels and also normalizing them. He implemented code for Hu's 7 moments algorithm along with normalization and obtaining dataset of features and output vector for Machine Learning models. He implemented Support Vector Machine for multi-class classification. M. V. Mani Kumar implemented code for Canny Edge Detection algorithm. He implemented Random Forest Classification for multi-class classification and Decision Tree Classification for multi-class classification.

REFERENCES

[1] M. Das, R. Manmatha, and E. M. Riseman, *"Indexing flower patent images using domain knowledge", IEEE Intell. Syst., vol 14, pp. 24-33, 1999.*

[2] T. Saitoh and T. Kaneko,*"Automatic recognition of wild flowers", System and Computer in Japan, vol. 34, no. 10,2003.*

[3] M. E. Nilsback, and A. Zisserman,*"A visual vocabulary for flower classification", In CVPR, 2006.*

[4] Y. Chai, *"Recognition between a large number of flower species", Master thesis, University of Oxford, 2011*

[5] Tanakorn Tiay, Pipimphorn Benyaphaichit and Panomkhawn Riyamongkol,*Flower Recognition System Based on Image Processing, In Third ICT International Student Project Conference,2014*

[6] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing Fourth Edition-Pearson Publicatons*

[7] "Flower Dataset", [online], Avaliable:*https://www.robots.ox.ac.uk/ vgg/data/flowers/17/*

[8] "Impoting All Images":*https://stackoverflow.com/questions/33369832/read-multiple-images-on-a-folder-in-opencv-python/33371454*

[9] "BGR to RGB":*https://www.geeksforgeeks.org/convert-bgr-and-rgb-with-python-opencv/*

[10] "Display of Image":*https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.imshow.html*

[11] "Resize":*https://www.geeksforgeeks.org/python-pil-image-resize-method/*

[12] "Grayscale":*https://en.wikipedia.org/wiki/Grayscale*

[13] "RGB to HSV":*https://scikit-image.org/docs/dev/auto_examples/color_exposure/plot_rgb_to_hsv.html*

[14] "Histogram":*https://numpy.org/doc/stable/reference/generated/numpy.histogram.html*

[15] "Maximum":*https://numpy.org/doc/stable/reference/generated/numpy.amax.html*

[16] "Minimum":*https://numpy.org/doc/stable/reference/generated/numpy.amin.html*

[17] "Empty Array":*https://numpy.org/doc/stable/reference/generated/numpy.empty.html*

[18] "Display of Grayscale Image":*https://stackoverflow.com/questions/12760797/imshowimg-cmap-cm-gray-shows-a-white-for-128-value*

[19] J. Canny,*"A computational approach to edge detection", IEEE trans. on pattern analysis and machine intelligence, vol. pami-8, no. 6, 1986.*

[20] "Canny Edge Implementation":*https://www.geeksforgeeks.org/implement-canny-edge-detector-in-python-using-opencv/*

[21] "Hu's moments":*https://en.wikipedia.org/wiki/Image_moment*

[22] "Balanced Data":*https://medium.com/analytics-vidhya/what-is-balance-and-imbalance-dataset-89e8d7f46bc5*

[23] "Data Split":*https://blog.roboflow.com/train-test-split/*

[24] "Range":*https://www.w3schools.com/python/ref_func_range.asp*

[25] "Random Sample":*https://www.geeksforgeeks.org/python-random-sample-function/*

[26] "Element in Element":*https://www.geeksforgeeks.org/python-check-if-element-exists-in-list-of-lists/?ref=lbp*

[27] "Nonselect":*https://www.w3schools.com/python/python_lists_comprehension.asp*

[28] "Machine Learning":*https://www.youtube.com/watch?v=Ato-hcv0NK0*

[29] "Support Vector Machine":*https://towardsdatascience.com/multiclass-classification-with-support-vector-machines-svm-kernel-trick-kernel-functions-f9d5377d6f02*

[30] "Support Vector Machine Documentation":*http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html*

[31] "Random Forest Documentation":*http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html*

[32] "Decision Tree Documentation":*https://scikit-learn.org/stable/modules/tree.html*

[33] "K Nearest Neighbour Documentation":*http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html*

[34] "f1 score":*https://scikit-learn.org/stable/modules /generated/sklearn.metrics.f1_score.html*

[35] "Accuracy Score":*https://scikit-learn.org/stable/modules /generated/sklearn.metrics.accuracy_score.html*