Plant Disease Classification and Content Based Image Retrieval

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

B.Tech

in

Electronics and Communication Engineering

By

Abishek.R (108116004)

Ahamed Irfan Mohamed Nusri (108116006)

T Arunkumar (108116094)

ELECTRONICS AND COMMUNICATION ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
TIRUCHIRAPPALLI – 620015

June 2020

In memory of

All those who lost their lives to COVID'19

Let's work together for a better world.

BONAFIDE CERTIFICATE

This is to certify that the project titled **Plant Disease Classification and Content Based Image Retrieval** is a bonafide record of the work done by

Abishek.R (108116004)
Ahamed Irfan Mohamed Nusri (108116006)
Arunkumar T (108116094)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Electronics and Communication Engineering** of the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI**, during the year 2019- 2020.

Dr. R. Malmathanraj
Guide

Dr. P Muthuchidambaranathan
Head of the Department

Project Viva-voce held on _____

Internal Examiner

External Examiner

ABSTRACT

This thesis is an attempt to develop a Content Based Image Retrieval (CBIR) system to retrieve images of diseased leaves of tomato plant. It uses colour, shape and texture features of the tomato leaf to classify and retrieve similar images. HSV colour histogram is used to extract colour features. Fourier descriptors provides shape feature, in the form of contour of the region of interest. Local Binary Pattern (LBP) is widely used for texture extraction. In order to consider global texture features, a variant of LBP called Completed Local Binary Pattern (CLBP) is utilized. Here we find sign component and magnitude component using the differences between the neighbouring pixels and its centre pixel. Also Uniform rotational invariant LBP is applied to reduce the number of patterns. Furthermore feature fusion of all colour, shape and texture properties is done to increase accuracy. Based on this feature vector, classification of disease is done using a supervised learning technique called Support Vector Machine (SVM). Analysis of different kernels like linear, RBF, polynomial etc. and hyperparameter optimization showed that Linear kernel is best suitable. Different combinations of features and their corresponding accuracy is found to choose the best accurate model. Similar analysis is carried out to find the suitable distance metric for retrieval purposes. In regards of classification, mean accuracy of 97.3% is achieved in linear SVC model by 5-fold cross validation. And in retrieval, mean average precision of 85.08% and Bull's eye performance of 90.17% are obtained using canberra distance metric.

Keywords : Colour Histograms, Fourier Descriptors, Local Binary Patterns, Completed LBP, Support Vector Machine, Content Based Image Retrieval

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to the following people for guiding us through this course and without whom this project and the results achieved from it would not have reached completion.

Dr.R.Malmathanraj, Assistant Professor, Department of Electronics and Communication Engineering, for helping us and guiding us in the course of this project. Without his guidance, we would not have been able to successfully complete this project. His patience and genial attitude is and always will be a source of inspiration to us.

Dr.P.Muthuchidambaranathan, the Head of the Department, Department of Electronics and Communication Engineering, for allowing us to avail the facilities at the department.

We are also thankful to the faculty and staff members of the Department of Electronics and Communication Engineering, our individual parents and our friends for their constant support and help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 General

Agriculture industry is one of the most vital industry in India which hugely contributes to the economy of nation. Agriculture sector is one of the least touched upon sector with technology, major part of the process is done physically by human. As the technology developed over the time period, many agriculture components and processes have become automated to ensure faster production and to ensure products are of highest quality standards. Because of the increased demand in the agricultural industry, it is vital that agriculture produce is cultivated using an efficient process [1]. Diseases and defects found in plants and crops have a great impact on production in the agriculture industry, and lead to significant economic losses [2]. Insect infestation along with bacterial, fungal, virus infection, changes in climate and temperatures are some of the few factors which mainly contributes to the diseases found in plants.

Tomato is one of the most caring food crops of India. This plant grown in 0.458 M ha area with 7.277 M mt production and 15.9 mt/ha productivity. It is cultivated throughout the year, but typically in summer and winter seasons. It is well grown in an average monthly temperature range of 21°C - 23°C but commercially it may be grown at temperatures ranging from 18°C to 27°C. Temperature and light intensity affect the pigmentation, fruit-sets and nutritive value of the fruits. Due to all these environments plant become very susceptible to diseases caused by fungi, bacteria, and viruses [3]. Tomato industry is back bone of the agriculture sector, so it is very much necessary to identify the diseases and take proper treatments at early stages. The most common method used by the farmers for the leaf disease detection is by naked eye. This requires adequate experience, continuous monitoring of the the plant, which might be very costly and time consuming in large farms. Another method used by the farmers is refer experts, which is expensive and not an option which is viable always. Therefore, detection and classification of diseases is an important and urgent task.

Bacterial spot, Septorial spot, Late blight and Tomato mosaic are the diseases which is included in our study. Bacterial spot disease is caused by the bacterium Xanthomonas vesicatoria, which attacks green but not red tomatoes. Septorial spot is destructive disease of tomato foliage, petioles and stems (fruit is not infected) is

caused by the fungus Septoria lycopersici. Infection usually occurs on the lower leaves near the ground. Late blight is a potentially serious disease, caused by the fungus Phytophthora infestans. Tomato mosaic virus causes yellow mosaic symptoms on the leaves and tomato fruits.

(a) Bacterial Spot (b) Septorial spot (c) Tomato mosaic (d) Late blight

Figure 1.1: Tomato leaf diseases

As the diseases are visible on the leaf itself, it is possible to detect it from the outer surface itself, instead of study of plant DNA. This project work is going to be detection of diseases from their leaves and classify it by extensively studying the disease symptoms on leaves. The symptoms are in the form of various visible patterns or holes which are shown on Fig 1.1 which will be the target to detect, observe and classify. Image segmentation can help to classify the affected leaves from the non affected one [4]. All the diseases have common characteristics that they changes some morphological characteristics such as colour,shape and texture. These changes can be extracted through image segmentation and classification of different disease through Multi-class SVM algorithm.

1.2 Objectives

Our primary objectives are to develop a trained model which classifies whether the tomato plant leaf is healthy or not, it identifies the exact disease and when a query image is given, similar images are retrieved based on colour, shape and texture from the database.

In the case of classification, our goal is to extract the features of colour, shape and texture of the image and train a supervised machine learning model with multi class SVM algorithm to classify the diseases and identify it. In terms of performance metrics, our goal is to increase the accuracy, precision and recall than the existing methods. Hyper parameter tuning will be done and will corner out the most robust model which produces increased accuracy.

In terms of retrieval, our goal is to propose a novel method to develop a content based image retrieval system which retrieves the similar images for the given query image from the database. In terms of performance metrics, our goal is to increase the retrieval accuracy, precision and recall than the existing methods. we explore varieties of similarity measurement metrics with different combination of features. Bull's eye performance and mean average precision will be done and will corner out the most robust similarity measurement.

The motivations of this project is to eliminate the need of monitoring the tomato plant leaves with naked eyes to detect the diseases instead create a system that detects the disease automatically and help out the farmers by saving their plant, time, money and energy.

CHAPTER 2

LITERATURE REVIEW

2.1 Analysis of content based image retrieval for plant leaf diseases using colour, shape and texture features [5]

Content Based Image Retrieval (CBIR) is more reliable and effective technique to retrieve similar images, based on the query image using their visual features like colour, shape, texture etc. CBIR has many applications in computer vision and image information systems which include digital libraries, medicine research, bio-diversity systems, fingerprint detection etc. Another such application realm is agriculture, where CBIR can be used to detect diseases early in leaf, steam and fruit. This paper tries to develop such CBIR system to detect and retrieve diseased soybeans leaves. The work presented in this paper have been achieved using a feature fusion of colour, shape and texture of the soybean leaf.

In this paper, HSV colour histogram was used for colour features as it provides better results than RGB and YC_bC_r. Scale Invariant Feature Transform (SIFT) was utilized to extract shape features, which used number of matching key points as a resemblance measure. This paper also proposed a novel texture feature named Local Gray Gabor Pattern (LGGP) which is a mix of Local Binary Pattern (LBP) and Gabor features. SVD (Singular Value Decomposition) was computed to get a single value relevant to the query image and each database image. For ranking, all SVD values are listed in in ascending order and similar images are retrieved. Retrieval efficiency of about 96%, 68% and 76% have been achieved for soybean leaves infected by mosaic virus, septoria brown spot and pod mottle disease respectively. It also indicated that classification performance depends greatly with the choice of infected leaves used in database formation, quality of photos, prominence of disease symptoms and disease intensity.

2.2 Apple disease classification using colour, texture and shape features from images [6]

In this study, an analytically verified method was used for the apple disease classification, using multiple type features such as colour, texture and shape. Defect

identification, feature extraction, feature fusion, training and classification was the flow of steps implemented. For accurate defect segmentation K-means clustering technique was done to get the region of interest and to remove the background. Various feature descriptors are used for feature extraction which includes: 1) Colour based - Global colour Histogram (GCH) and Colour Coherence Vector (CCV), 2) Texture based - Local Binary Pattern (LBP) and Completed Local Binary Pattern (CLBP), and 3) Shape based - Zernike Moments (ZM).

In this paper three kinds of apple diseases, including Apple Blotch, Apple Rot, and Apple Scab were considered for dataset. GCH is the list of values describing the probability of a pixel being of the identical colour. Colour coherence is the amount to which, the pixels of identical colour contributes to a part of huge uniform area. Zernike Moments was used due its properties of rotational invariant and orthogonality. Multi- class Support Vector Machine (MSVM), a supervised machine learning classifier was used for classifying the query image. Analysis of various combinations of feature sets showed that CCV + CLBP + ZM combination gave the best accuracy of 95.94%.

2.3 Color texture description with novel local binary patterns for effective image retrieval [7]

This paper proposed a new local color texture descriptor called local binary pattern for color images (LBPC). It is based on the concept of separating color pixels in an image using a hyper-plane in m number of dimensions. Three dimensional (m=3) hyper-plane was found to be effective than other threshold models like hyper-sphere, hyper-cube or hyper-ellipsoid. RGB model was used and the pixels were thresholded in a circular neighborhood of radius R and members P. By analyzing it was found out P=8 and R=2 gave better accuracy. LBPH which is local binary pattern for hue component was also proposed to increase accuracy.

This paper elaborates on the proposed five different modalities: LBPC, ULBPC, LBPH, ULBPH, CH, and their four combinations: LBPC+LBPH, ULBPC+ULBPH, LBPC+LBPH+CH, and ULBPC+ULBPH+CH. Further analysis revealed that LBPC + LBPH + CH combination resulted in the best retrieval performance. This combiantion also uses small number of features (542), which further results in accelerated retrieval. In regard of retrieval efficiency, extended-Canberra distance metric outperformed other metrics, to find similar images.

2.4 Tomato Plant Disease Detection using Image Processing [3]

This paper tries to utilize image processing techniques to classify diseases, in one of the most common food crop in India, tomato plant. It tackles four common early diseases namely Early blight, Iron chlorosis, Septoria leaf spot and Bacterial spot. The paper proposes a series of step to identify the disease starting with acquiring quality image dataset and smoothing them using kurtosis and skewness filters. The infected parts of the leaves were segmented utilizing the inverse difference method. Then distinctive features were extracted from these segmented images. For colour features, co-occurences matrices were found out by converting RGB to HIS colour space and mapping each pixels. This paper also proposes four properties namely Energy, Entropy, correlation and homogeneity to extract texture features. For classification, Multi-class Support Vector Machine model is used to train and test, using 80 images of each disease. This proposed methodology was able to sort the diseased plant with 93.75% accuracy.

2.5 Analysis of Different Similarity Measures in Image Retrieval Based on Texture and Shape [8]

This paper explores various similarity measures and its effectiveness in CBIR systems. Only texture and shape features were extracted and combined to form the feature vector. LBP was used to extract the texture features, which gives the structural information of objects in an image. To retrieve shape features, Edge histogram descriptor (EHD) was proposed which also gives geometric information about colour. It fundamentally shows the frequency of appearance of five cases of edges, in every local area called a sub-image or image block. The sub-image is formed by partitioning the image space into 4x4 non-overlapping blocks. After concatenating both the features, similarity measurement was done by implementing various distance metrics. This includes Minkoswki Distance, Euclidian distance, Manhattan Distance and Spearman rank coefficient. WANG database was used to test this proposed system, which had six general classes namely Africans, Beach, Architecture, Buses, Dinosaurs and Horse. To evaluvate and compare performance for each metric, precision and recall were found out. Top 40 similar images were retrieved based on the input query image. The paper, by analyzing the results concluded that Spearman rank coefficient produced better result in comparison with other distance metrics.

CHAPTER 3

METHODOLOGY

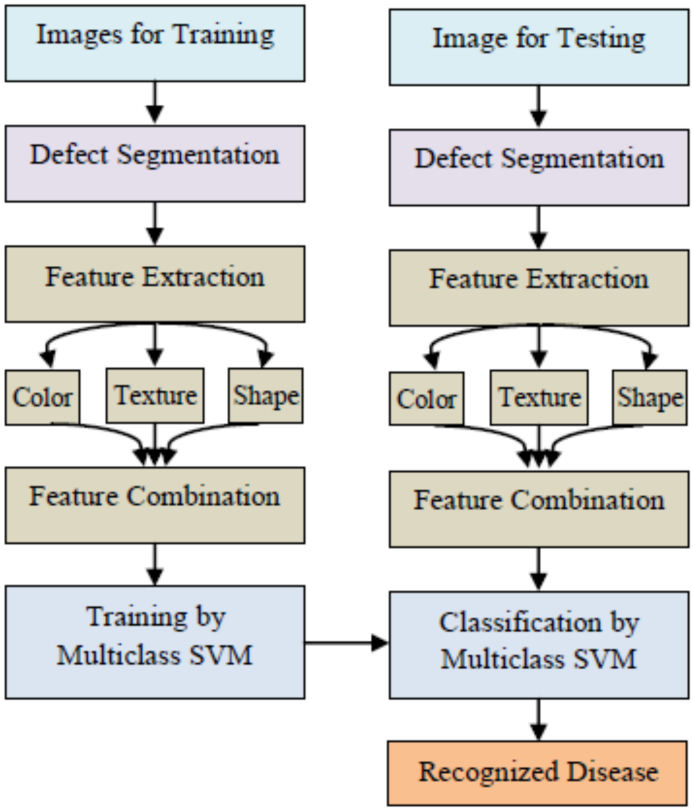


Figure 3.1: Workflow flow disease classification

Our goal is to classify the tomato leaves diseases and retrieve similar images from the database when a query image is given. For this we have obtained several images from each category of diseases namely: Bacterial spot, Septorial spot, Late blight, Tomato mosaic and healthy tomato leaves. Classification of diseases is achieved according to the workflow chart given in Fig 3.1. Initially colour, texture and shape features are obtained for each images and those features are combined and feature vector is created. For classification 70% of the images are used for training the multi class SVM model and remaining 30% are used testing. Accuracy of the model is determined by K-fold cross validation of the trained model.This model is saved for further classification of new images. In the case of retrieval, combined feature vectors of query image is compared for similarity by using distance metrics such as Canberra, Euclidean, Manhattan and Chebyshev with the feature vectors in database. Fig 3.2 show the process of comparing two images. Most similar images

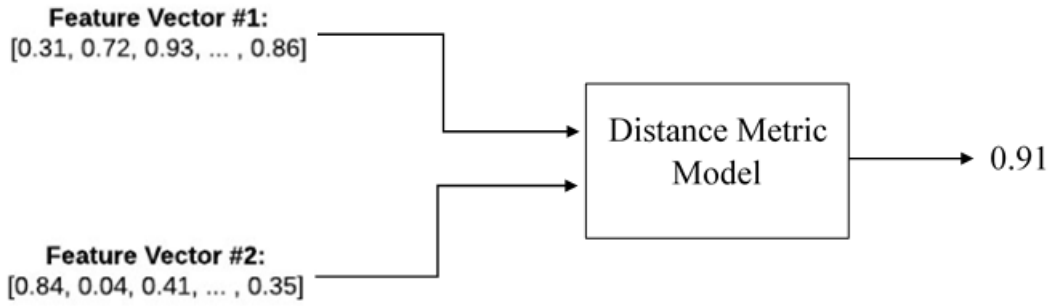


Figure 3.2: Process of distance metric model

which have lowest distances comparable to query image is displayed. Given two feature vector, a distance metric model is used to find how similar the two feature vectors are, and according to that similar images will be retrieved.

3.1 Pre-processing

3.1.1 Resizing

The images considered are resized to standard dimensions to ensure uniformity. They are resized to height of 800 pixels and width of 533 pixels.

3.1.2 Segmentation

A typical image consists of regions of interest (ROI) along with background noise. It is necessary to select only required information and suppress other undesirable components. This will result in lesser interference in feature extraction in ensure cleaner feature vectors.

Figure 3.3: Original image

Image segmentation is performed to select the ROI by K-means Clustering method [9]. K-means Clustering is an unsupervised machine learning to group similar data points together. A target number k is defined which represents the number of centroids or cluster centers.

Algorithm

- The algorithm starts by selecting 'k' centroids randomly.
- The distance between each data point and each of centroid is calculated.
- A particular data point is allocated to that centroid to which distance is minimum when compared to other centroids.
- The new centroid location is calculated by considering the data points which are considered to be associated with that centroid.
- Distance between data points and new centroids are found and reassigned.
- The process is repeated until no data points are reassigned or maximum number of iterations set are reached.

Applying the algorithm to images present in data set with 'k' = 2 gave satisfactory results in selecting only sections that contained leaf while the background section got clustered separately.

Figure 3.4: Segmented Image

3.1.3 Bilateral filtering

Blurring, also called smoothing, is a frequently implemented image processing technique. It is done for many reasons, but mainly for noise removal. Filters are

like a window of co-efficents, sliding across a picture. There are different filters present like Gaussian, Median, Box to smoothen the image. However certain times, these filters will not only remove the noise but also smoothen the edges. In order to avoid this, a non-linear and edge-preserving smoothing filter called Bilateral filtering is utilized.

The fundamental idea behind bilateral filtering is to do in the scope of an image, what normal filters do in its domain. Two different pixels can occupy nearby values and nearby spatial locations. First, shift-invariant low-pass filter is implemented. In order to maintain the geometric and photometric locality, range filtering is applied. So fundamentally, bilateral filtering is combination of domain and range filtering along with normalization. It changes a certain pixel value with the mean of nearby and close values. In smooth regions bilateral filter simply acts as domain filter, since pixel values in such neighbourhoods are very much alike. In boundary between dark and bright regions, bilateral filter makes sure it doesn't blur it. Because in bright spots, the filter will ignore the dark pixels and will consider only nearby bright pixels for taking mean. Similarly in dark spots, the filter will ignore the bright pixels and will consider only nearby dark pixels for taking mean. So through this method, it preserves the edge and removes noise in other areas.

3.2 Feature Extraction

Features are measurable properties or characteristics of data that is being analyzed. For better classification and retrieval it is crucial to select independent features that are discriminating and informative. In the proposed methodology colour, shape and texture features are extracted from images.

3.2.1 Colour

Color features are the fundamental feature of the content of images. Through it, we humans can detect most pictures and objects present in it. Different methods like colour histogram, colour correlogram, color moments and color structure descriptor are used to extract these straightforward features. In this proposed system, colour histogram is implemented. Histogram constitutes image from a different perspective. It gives the frequency distribution of colours i.e. the number of times each colour has appeared in the image pixels. Colour histogram is rotation invariant on the view axis and only changes slightly if rotated otherwise. This robustness and its easy to implement simplicity, is what makes histogram a widely used colour feature extractor. Also the size of the histogram is very much smaller than that of the corresponding image. Global color histogram and local color histogram are the two types. Generally, a common a colour space is used to comprise all the colours of

an image. A three dimensional system is used to describe such a colour model, where each point in the space will signify a colour. RGB (red, green, blue), HSV (hue, saturation, value) and YC_bC_r (luminance and chrominance) are commonly used models in image processing.

The primary colours Red, Green, Blue are additive and new colours can be formed by varying the intensity of each channel of RGB. This is the general perception of colours by us, humans. Another model called HSV is sourced from RBG space, which has the RGB's main diagonal as its vertical axis. Saturation can take values from 0 (gray) to 1.0 (no white component). Red, yellow, green, cyan, blue and magenta and back to red are the colours obtained when Hue is modified from 0 to 360 degrees. It is easier to separate the colour luminance information, in other words intensity, from the HSV model but it is not possible in RGB model. Often we need this intensity values to work with, in image processing applications. So HSV colour space model is widely preferred in such image processing systems [10].

For the three channel image, the histogram is generally split into bins in order to crudely represent the image and to reduce dimensionality. Then by directly combining these three histogram, single feature vector can be obtained. Sometimes histograms are normalized so that the sum of all its values equals to one. This feature vector can itself be used to retrieve similar images to query image. But accuracy will not be high enough for complex images. So further more distinctinve features are extracted and concatenated.

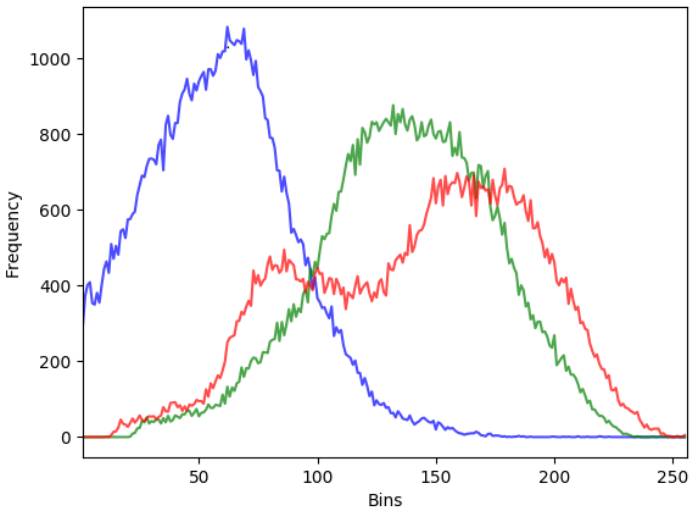


Figure 3.5: RGB histogram of a bacterial spot affected leaf

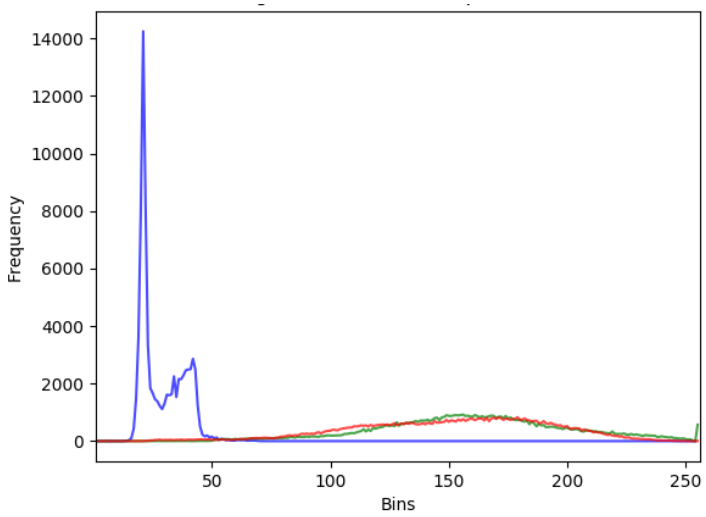


Figure 3.6: HSV histogram of a bacterial spot affected leaf

The above figures shows the histogram of bacterial spot affected tomato leaf in RGB and HSV colour space. In HSV the feature size is reduced from 768 to 692.

3.2.2 Shape

Shape features are set of numbers which effectively represent the boundary of image. There are several methods which are known for encoding shapes like minimum bounding rectangle, polygon evolution, convex hull, SIFT etc. They can be classified based on method extraction like contour based or region based. We propose to use Fourier descriptors to extract shape features.

Fourier descriptors are contour based approach. Each point in the image contour is converted to an imaginary number x + iy from x , y plane for 1D representation. Fourier transform is then performed on each point. This resulting Fourier coefficients obtained is considered as feature vector. But the feature length is in order of thousands and variable for different images. To ensure uniformity and reduced feature length, M coefficients from total N coefficients are considered. The M lower frequency components approximately describes shape.

For this approach a binary image which contains the leaf contour is given as input.

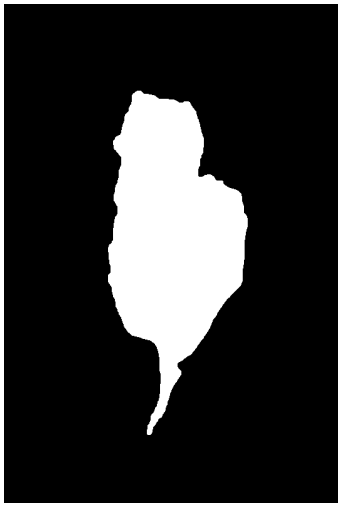


Figure 3.7: Binary image with leaf contour

For m[th] contour out of N boundary points in contour z[m] represents it's imaginary form.

$$z[m] = x[m] + iy[m] \tag{3.1}$$

Descriptor is found by computing discrete fourier transform of z[m].

$$Z[k] = DFT[z[m]] = \frac{1}{N}\sum_{m=0}^{N-1} z[m]e^{\frac{-j2\pi mk}{N}} \quad (k = 0, 1, ...N-1) \tag{3.2}$$

The zero frequency component is center shifted and M where M<N components in the middle is truncated. To reconstruct the boundary inverse discrete fourier transform is performed over M coefficients.

$$\tilde{z}[m] = IDFT[Z[k]] = \sum_{k=-M/2}^{M/2} Z[k]e^{\frac{j2\pi mk}{N}} \quad (m = 0, 1, ...N-1) \tag{3.3}$$

The reconstructed image is shown here by taking M = 30.

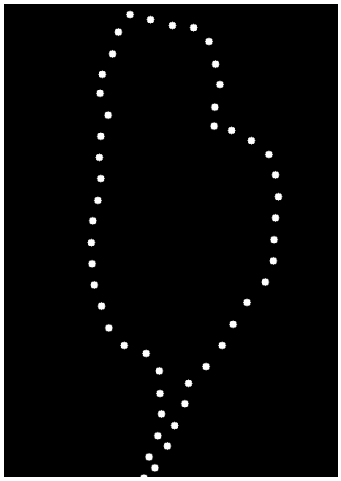


Figure 3.8: Reconstructed image

Shape descriptors are expected to be invariant of translation, scale, rotation and starting point. Fourier descriptor coefficients are minimally modified like change of DC component, scaling by same factor, rotation by same factor, and time shifting respectively [11]. The amount of modification is significantly lesser when compared to other techniques. This makes fourier descriptors a suitable candidate for shape feature extraction.

3.2.3 Texture

Local Binary Pattern (LBP)

Local Binary Pattern (LBP) is an approach to extract local features from image by computing the local differences in intensity between the value of the center pixel and its surrounding pixel (neighbouring pixels). The aim of the local binary pattern is detecting different patterns using the differences between the neighbouring pixels and its centre pixel. 8 Patterns are then aggregated to describe the whole image. Consequently, LBP have been used in enormous applications such as texture classification [12], face recognition [13], fingerprint identification [14] and medical image classification [15]. Here LBP descriptor is applied to gray scale images, so that each pixel has a value ranging from (0 to 255). In order to reduce the noise in the image, Bilateral filter has been applied prior to the image . The term LBP P, R is referring to original LBP descriptor where R is the radius of the circle and P is the number of neighbor pixels. Subsequently, the form of LBP 8, 1 represents the simple form of the LBP with radius one and eight neighbors. Thus, in a simple form, the local binary pattern generates binary code with eight bits, where each bit has the value of either one or zero. The value one is assigned if the neighbor of the center pixel has a greater grey value than the center pixel and zero otherwise. The eight neighbors around the centre can be described with a code of eight bits.

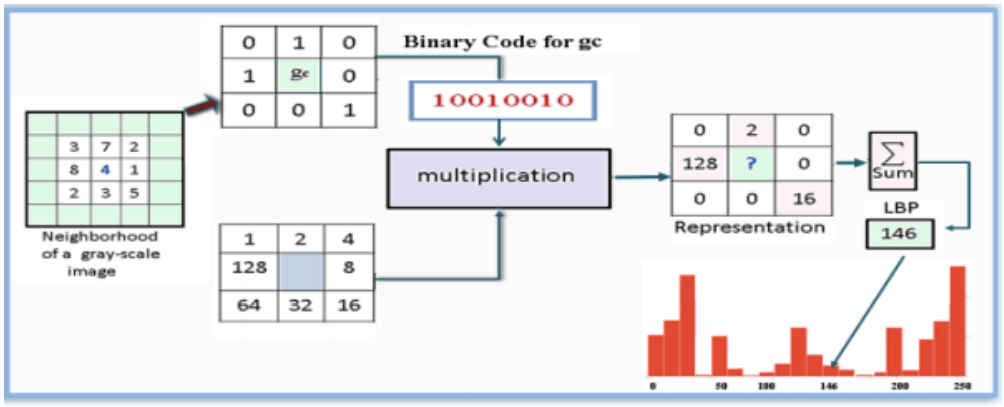


Figure 3.9: How LBP descriptor works

Fig 3.9 shows an example of the method of calculating the LBP descriptor.

$$S(x) = (W_c, W_0 - W_c, W_1 - W_c, ...W_{p-1} - W_c) \tag{3.4}$$

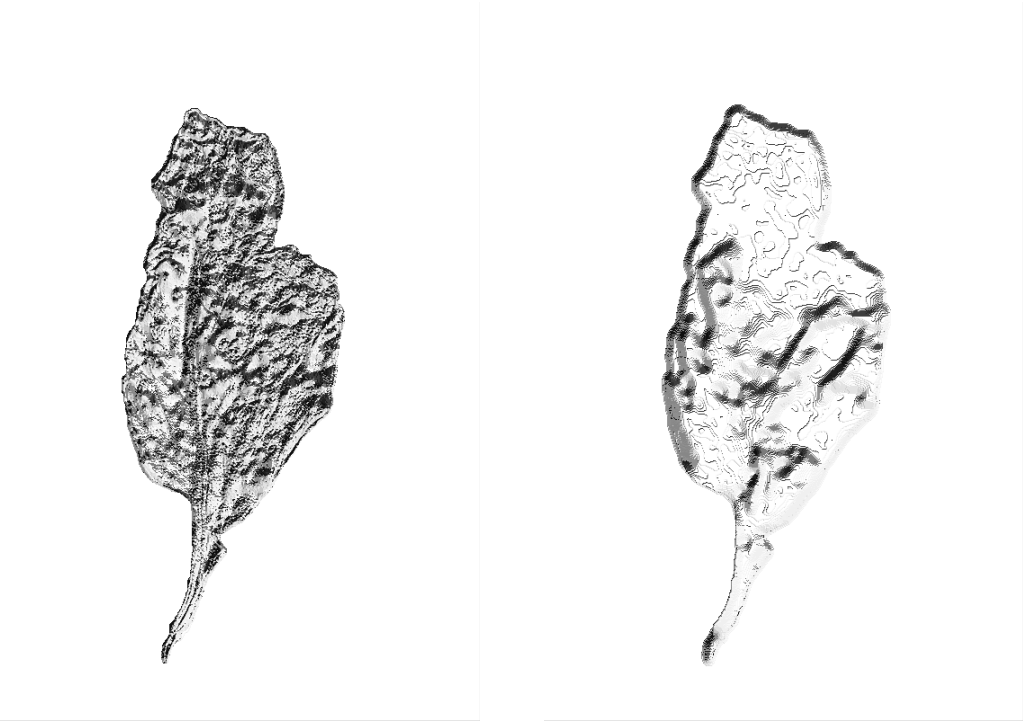
where W_p and W_c are neighbouring pixel and central pixel.

$$S(x) = \begin{cases} 1, & if \quad x \geq 1 \\ 0, & otherwise \end{cases} \tag{3.5}$$

The mathematical equation can be defined as:

$$LBP_{P,R} = \sum_{p=0}^{p-1} S\left(W_p - W_c\right) \times 2^p \tag{3.6}$$

LBP is applied to gray scale image and hue colour component of HSV colour space of Fig 3.4 and the output is given in Fig 3.10. LBP is applied on hue colour plane in HSV colour space to also include the colour component.



(a) Gray scale image (b) Hue component of HSV

Figure 3.10: LBP applied images

Histograms are computed for both gray-scale and hue colour spaces to compare their performance. Fig 3.11 shows how the histogram is constructed. Fig 3.12 show the histogram of Fig 3.10

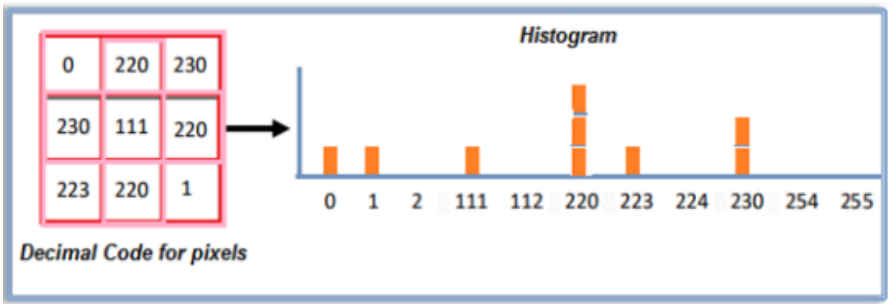
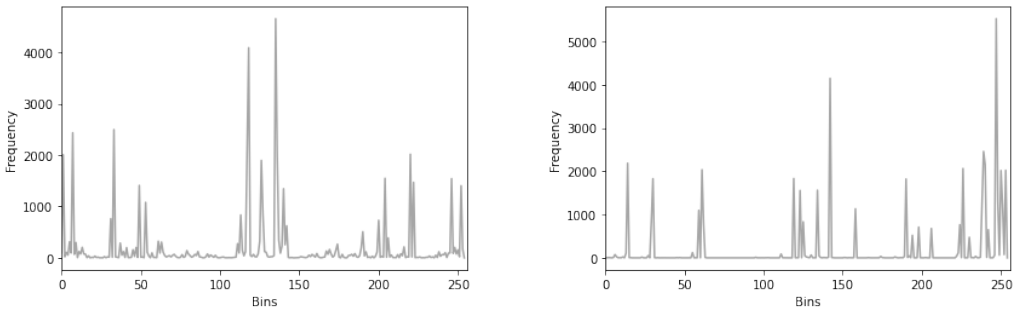


Figure 3.11: Example of histogram calculation

Since LBP has some limitations such as they produce rather long histograms, which slows down the recognition speed especially on large-scale face database, Under some certain circumstances, they miss the local structure as they don't consider the effect of the centre pixel, they are sensitive to noise, small spatial support area. As these are some concerns in the case of the retrieval and classifying efficiency, in order to increase the efficiency we have used a modified LBP called as Complete Local Binary Pattern.



(a) Gray scale component (b) Hue component

Figure 3.12: Histograms of LBP

Completed Local Binary Pattern (CLBP)

CLBP is a more complete manner to identify the texture of the image. Typical Local binary Pattern operator only considers the local features of an image and global features were neglected. To improve the performance of LBP by considering global thresholding and a new approach named as Complete Local Binary Pattern is proposed by [16]. Here we find 2 components: sign component and magnitude component. Magnitude component is calculated by the difference of central pixel and neighbouring pixels. The difference is compared with the global mean and converted to a 8-bit value. Sign and magnitude component can be calculated as shown in below equation:

$$S_p = S(W_p - W_c) \tag{3.7}$$

$$M_p = |W_p - W_c| \tag{3.8}$$

Where S_p is sign component and M_p is magnitude component. S_p and M_p are used to find the CLBP_S and CLBP_M operators respectively. CLBP_S is calculated as same as finding LBP (3.10), while in CLBP_M the local variance of the magnitude is calculated(3.12). These two components can be described mathematically as follow:

$$CLBP\_S_{P,R} = \sum_{p=0}^{p-1} 2^p \, S(W_p - W_c) \tag{3.9}$$

$$S_p = \begin{cases} 1, & W_p \geq W_c \\ 0, & W_p < W_c \end{cases} \tag{3.10}$$

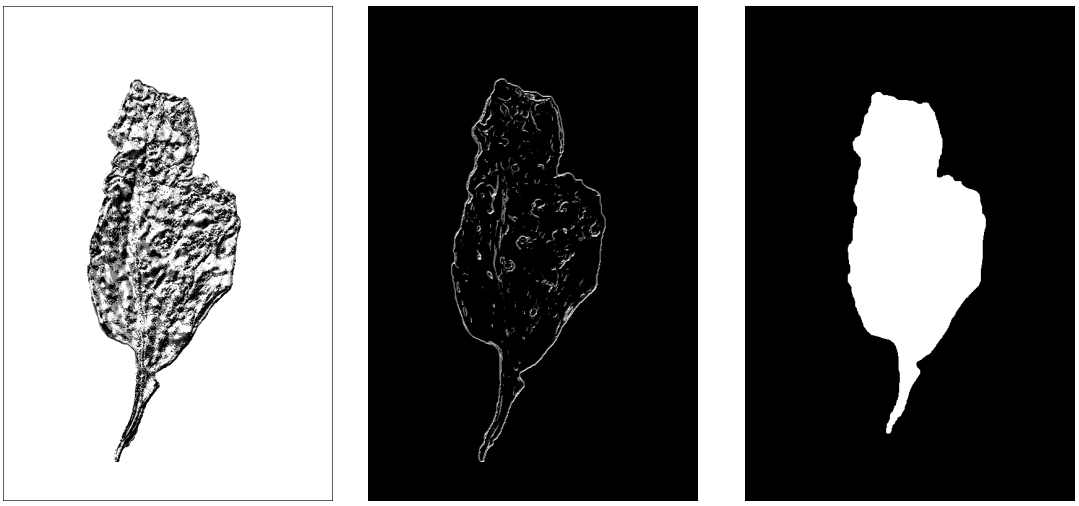
$$CLBP\_M_{P,R} = \sum_{p=0}^{p-1} 2^p \, t(M_p, C) \tag{3.11}$$

$$t(M_p, C) = \begin{cases} 1, & |W_p - W_c| \geq C \\ 0, & |W_p - W_c| < C \end{cases} \tag{3.12}$$

Where W_p and W_c are neighbouring and central pixel respectively, and c is the mean value of magnitude component in whole image.As the limitation of LBP is that it doesn't take into account of the effect central pixel, in CLBP it has been rectified by identifying the CLBP_C operator, which is calculated as below:

$$CLBP\_C = t(W_c, c_i) \tag{3.13}$$

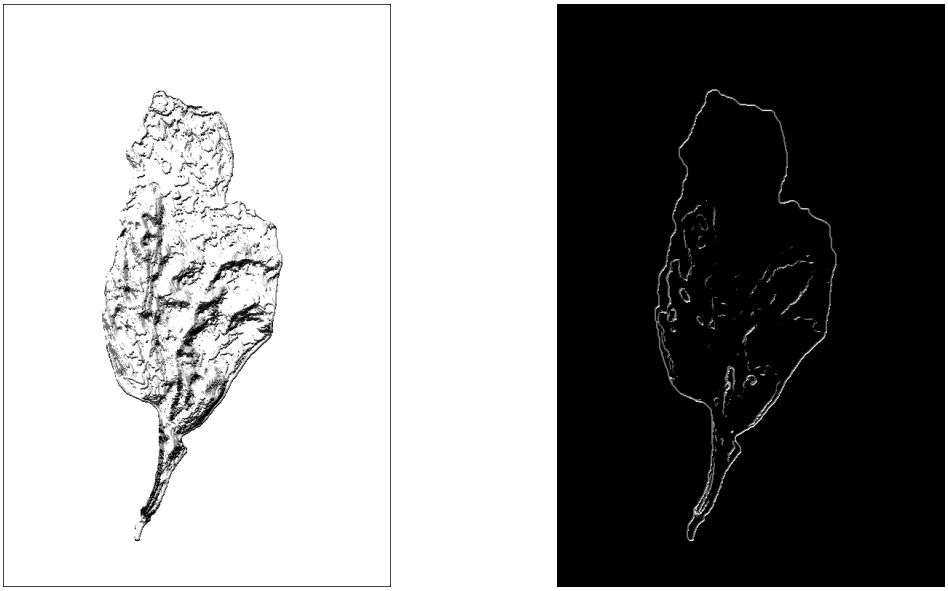
Where W_c is the center pixels value and C_i is the average gray level of the whole image. CLBP is applied on hue colour plane in HSV colour space to also include the colour component and for gray scale image. Figures below depict the obtained CLBP operators for the gray scale image 3.13 and hue colour component 3.14 for Fig 3.4.



(a) CLBP_S (b) CLBP_M (c) CLBP_C

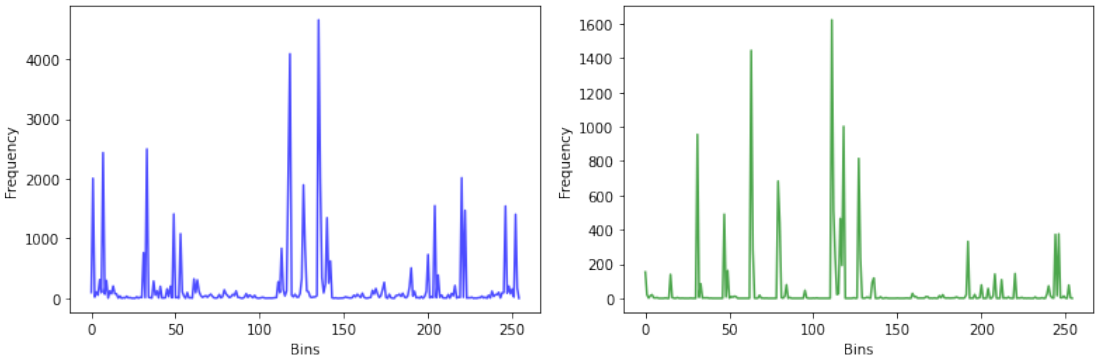
Figure 3.13: CLBP for gray scale of the image

Then, CLBP_S and CLBP_M components are joined together linearly and feature vector is obtained. Histograms are computed for both gray-scale and hue colour spaces to compare their performance. Since P=8 in our case, it will generate 2^8 = 256 histogram bins. Therefore, each bin of the histogram can be regarded as a "micro-texton" encoded by LBP [13].



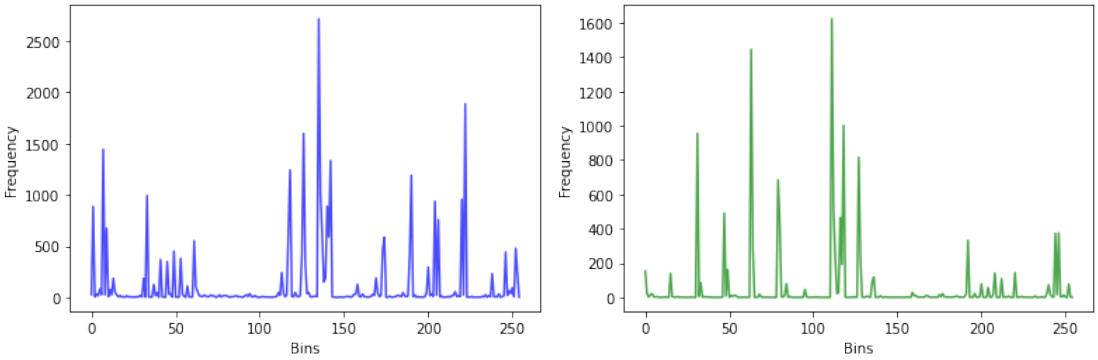
(a) CLBP_S (b) CLBP_M

Figure 3.14: CLBP for hue color component of the image



(a) CLBP_S (b) CLBP_M

Figure 3.15: Histogram of CLBP for gray scale image



(a) CLBP_S (b) CLBP_M

Figure 3.16: Histogram of CLBP for Hue colour component of HSV colour space

As we can see since the CLBP and LBP histogram lead to 256 patterns genera-

tion, the descriptor might be comparatively slow. So if we could reduce the pattern, it makes the CLBP descriptor faster to be generated and more robust in detection of primitive's properties in the image. Moreover, dealing with reduced number of patterns becomes easier and more flexible than normal CLBP without losing information about the image primitives. This can be done by uniform rotation invariant method.

Uniform Rotation Invariant

Binary codes which are created by LBP represents the pattern in the image. 3.17 illustrates the different patterns in a image.

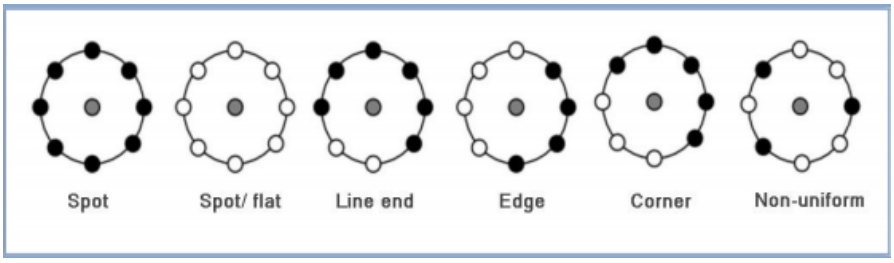


Figure 3.17: Image primitive captured by LBP patterns

Generally LBP patterns are divided into two categories: Uniform and Non Uniform pattern. Patterns are identified as uniform if the number of transition from binary 1 to binary 0 or vice versa is not more than two. If there exist more than two transition, then it is known as Non uniform pattern. Uniform patterns considered more influential than the others and it can be calculated as shown in equation (3.14) [17].

$$ULBP_{P,R} = |S(W_{P-1} - W_C) - S(W_0 - W_C)| + \sum_{P=1}^{P-1} |S(W_P - W_C) - S(W_{P-1} - W_C)| \tag{3.14}$$

Uniform rotational invariant LBP is applied to both gray scale and hue colour component for the 3.4. Fig 3.18 and Fig 3.19 show the output of uniform rotational invariant LBP.

(a) URI_S (b) URI_M

Figure 3.18: Uniform Rotational Invariant LBP for gray scale image

(a) URI_S (b) URI_M

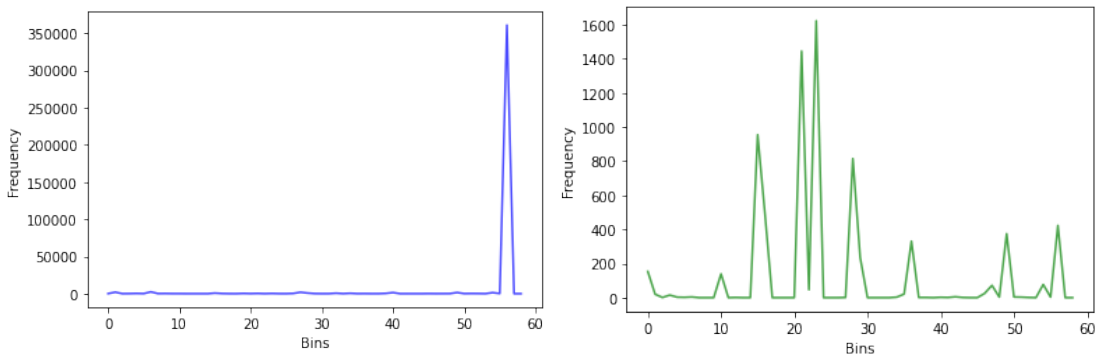
Figure 3.19: Uniform Rotational Invariant LBP for hue colour component

So according to uniform method, it produces P(P-1)+2 uniform patterns, where P is the number neighbour pixels. Hence the number of uniform LBP in our case is 58, and all the non uniform patterns are accumulated into one bin. So the total number of bins in a histogram in our case is reduced to 59. P(P-1)+3 gives the total number of bins in the uniform method. Table 3.1 shows the example of the Uniform and Non uniform LBP patterns.

Circular Binary Pattern	Bit wise Transition	Uniform Pattern
11111111	0	Yes
00001111	1	Yes
11001110	3	No
10101010	8	No
11111101	2	Yes
01001011	6	No
11000000	1	Yes

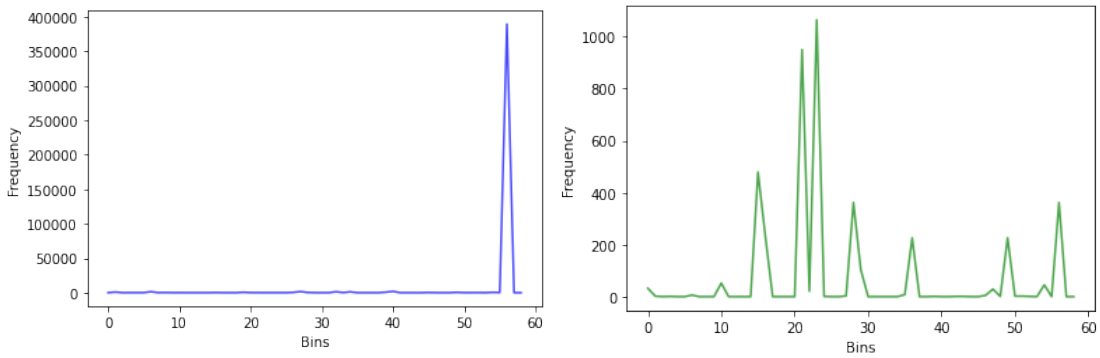
Table 3.1: Uniform and Non Uniform patterns

Histogram is applied for Fig 3.18 and Fig 3.19. Figure below shows the histogram for the above image.



(a) URI_S (b) URI_M

Figure 3.20: Histogram of uniform rotational invariant for gray scale image



(a) URI_S (b) URI_M

Figure 3.21: Histogram of uniform rotational invariant for hue colour component of HSV colour space

3.3 Feature Combination

The feature vectors obtained from colour, shape and texture properties are concatenated linearly to obtain a single feature vector for a particular image in database. This process is repeated for every image in database to obtain a feature bank for classification and retrieval purposes.

3.4 Disease Classification

Given an image of a leaf, a classifier has to be built to find if it is healthy or not. Also the exact disease should be identified. This can be achieved using several machine learning algorithms. We have used multi-class support vector machine(SVM) to achieve the same.

3.4.1 Overview of SVM

Support Vector machine identifies the decision boundary that separates different classes and maximises the margin. Margin is defined as the perpendicular distances between the decision boundary and the data points closest to it. The data points can be in n-dimensional space and the decision boundary is called a hyperplane and is of n-1 dimensions. Hyperplane is computed by considering the extreme points of a class and they are called as support vectors, hence the name support vector machine.

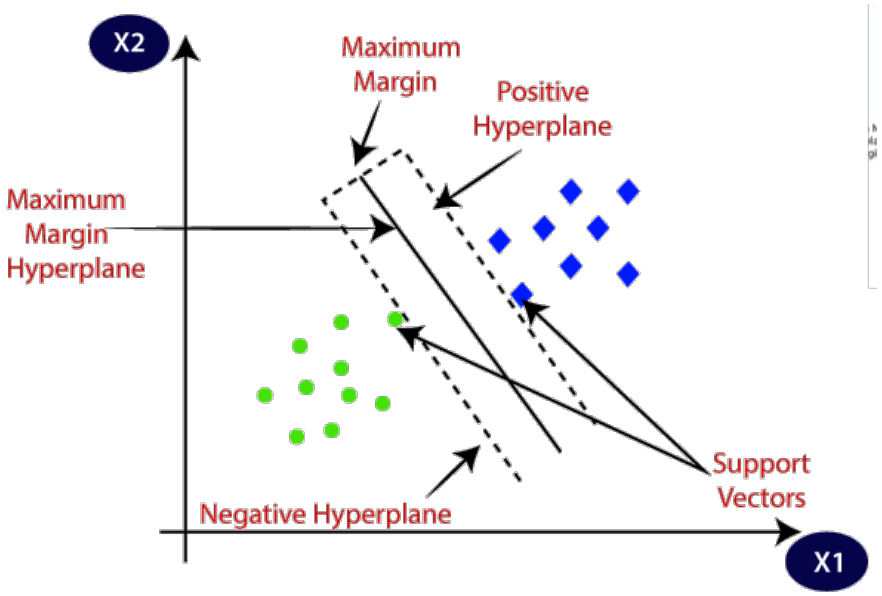


Figure 3.22: Support Vector Machine

But if data-points are not linearly separable like in 3.23, optimal hyperplane is computed by a method named 'kernel trick'.

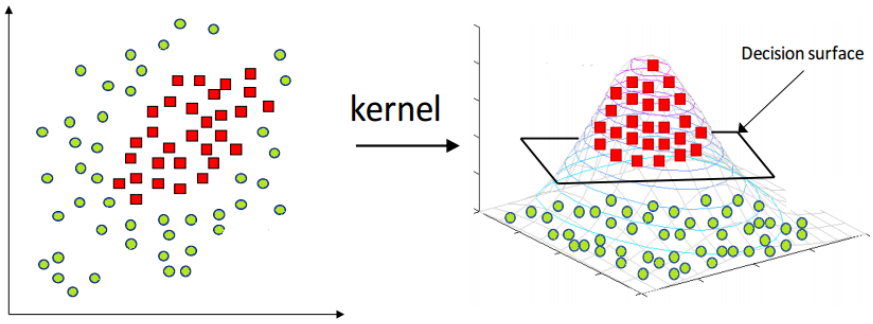


Figure 3.23: Kernel Trick

Non linearly separable data is mapped to higher dimension, to find a decision boundary. But actual transformation in case of higher dimensions is computationally

intensive. The kernel trick allows to operate in original feature space without transformation to higher dimensions.

For instance, two data-points x , y in 3 dimensional space is to be mapped to a 9-dimensional space.

$$\mathbf{x} = (x_1, x_2, x_3)^{\mathrm{T}} \tag{3.15}$$

$$\mathbf{y} = (y_1, y_2, y_3)^{\mathrm{T}} \tag{3.16}$$

The following operations have to be completed for transforming and the computational complexity is O(n^2)

$$\phi(\mathbf{x}) = ({x_1}^2, x_1x_2, x_1x_3, x_2x_1, {x_2}^2, x_2x_3, x_3x_1, x_3x_2, {x_3}^2)^{\mathrm{T}} \tag{3.17}$$

$$\phi(\mathbf{y}) = ({y_1}^2, y_1y_2, y_1y_3, y_2y_1, {y_2}^2, y_2y_3, y_3y_1, y_3y_2, {y_3}^2)^{\mathrm{T}} \tag{3.18}$$

$$\phi(\mathbf{x}_{\mathrm{T}})\phi(\mathbf{y}) = \sum_{i,j=1}^{3} x_{\mathrm{i}}x_{\mathrm{jij}} \tag{3.19}$$

3.4.2 Kernels

Employing a kernel function denoted as k(x,y), the same operation is completed in a 3 dimensional space while computational complexity is O(n).

$$k(\mathbf{x},\mathbf{y}) = (\mathbf{x}^{\mathrm{T}}\mathbf{y})^2 = (x_1y_1 + x_2y_2 + x_3y_3)^2 = \sum_{i,j=1}^{3} x_{\mathrm{i}}x_{\mathrm{jij}} \tag{3.20}$$

Kernel trick offers an efficient way to transform data and not limited to SVM. There are different kernels available:

- Linear
- Radial Basis Function
- Polynomial

Linear kernel is defined as

$$k(\mathbf{x},\mathbf{y}) = \mathbf{x}^{\mathrm{T}}\mathbf{y} + c \tag{3.21}$$

and is used in cases of linearly separable data.

Polynomial kernel not only looks for similarity in original features, but also in combination of them.

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^{\mathrm{T}}\mathbf{y} + 1)^{\mathrm{d}} \tag{3.22}$$

where d is the dimension.

Radial basis kernel is defined as

$$k(\mathbf{x}, \mathbf{y}) = \exp\{-\gamma \|\mathbf{x} - \mathbf{y}\|^2\} \tag{3.23}$$

γ defines how each training examples has influence in determining the hyperplane. Lower values indicates a larger influence from farther data points and larger values from closer points to hyperplane.

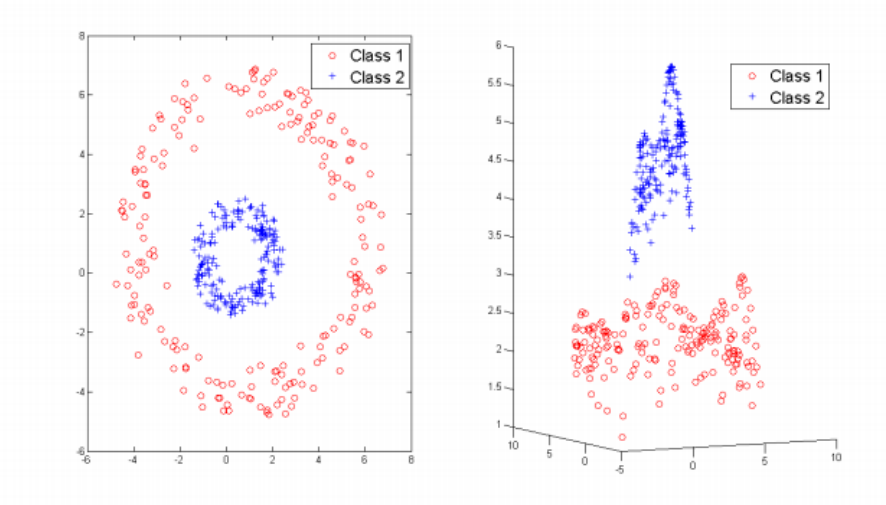


Figure 3.24: RBF kernel transformation

Hyperplane identification is also dependent upon regularization parameter C. For larger values of C, smaller margin hyperplane is chosen so training points are classified correctly. But a smaller C will cause result in larger margin, causing mis-classifications in test data set.

3.4.3 Hyper-parameter tuning

It is necessary to vary all the parameters like kernel, C and γ to check for highest accuracy. This method is called as Hyperparamter tuning.

Once the parameters which results in best possible accuracy is determined, the model is finalized and can be used for classification.

3.4.4 Evaluation Metrics

K-Fold Cross Validation

Once a model has been finalised, the performance of the model has to be evaluated. K-fold cross validation is a method of evaluation. K in K-fold determine the number of different groups of how data is split into. Then each group individually is

considered as test data while rest others are used for training the model. Evaluation performance is retained for each fold and averaged over K iterations. The mean accuracy determines how well the model can perform on unseen data based on limited given data.

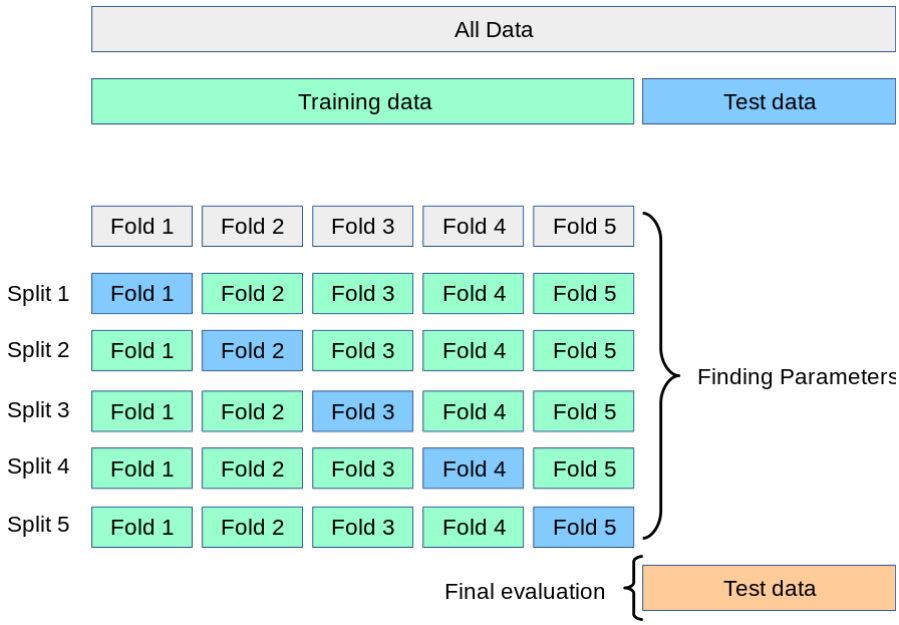


Figure 3.25: 5-fold cross validation

Confusion Matrix

Confusion matrix tests the performance of the classifier. Each predicted output class is plotted and counted against the actual class label.

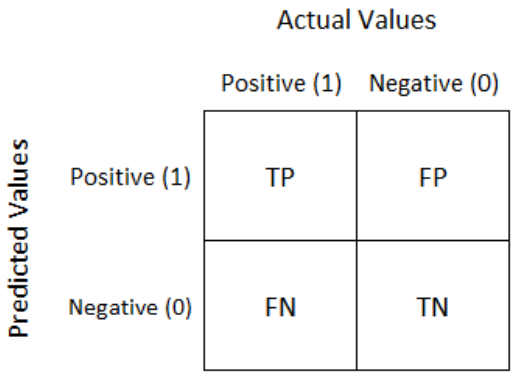


Figure 3.26: Confusion matrix for binary classification

Based on the predicted and actual output labels, each call is defined as

- **True Positive:** If a positive class is actually predicted positive
- **True Negative:** If a negative class is actually predicted negative

- **False Positive:** If a negative class is predicted positive
- **False Negative:** If a positive class is predicted negative

Precision and recall are computed based on the confusion matrix values.

$$Precision = \frac{TP}{TP + FP} \tag{3.24}$$

Precision tells about the amount of actual positive classes out of the predicted positive classes.

$$Recall = \frac{TP}{TP + FN} \tag{3.25}$$

Recalls gives information about how many positive classes are predicted correctly out of total positive classes.

A comparable measure is found by combining precision and recall value as F1 score.

$$F1 = (1 + \beta^2).\frac{precision.recall}{(\beta^2.precision + recall)} \tag{3.26}$$

β is defined as a positive real value where recall is β times more important than precision. In most cases β value of 2 is taken. The same concepts can be extended to multiclass classification problems as well.

3.5 Image Retrieval

CBIR(Content Based Image Retrieval) system has been used for the image retrieving purpose. The objective of content based image retrieval is, when an image of a leaf is given as query, system has to develop techniques to automatically extract and retrieve similar images from the huge database.

3.5.1 Overview of CBIR

The problem of searching similar images from large image repositories on basis of their visual contents is called content-based image retrieval [18]. The term content in Content Based Image Retrieval (CBIR) refers to colors, shapes, textures [19] or any other information that can be obtained from the image itself. There are two significant phases in the CBIR:

1. Indexing Phase: Where images features like colour, texture and shape are extracted from the image and they are consequently stored in index data.
2. Retrieval phase: Query image is given. Based on the similarity distance measure between query image and database images relevant similar images are retrieved.

Fig 3.27 gives the overall view of how CBIR system works.

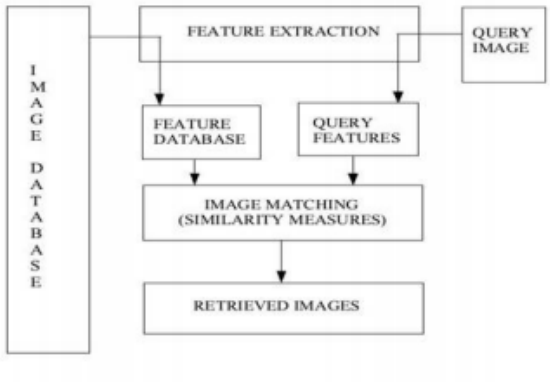


Figure 3.27: Architecture of CBIR

3.5.2 Approach For Image Retrieval

Approach for the image retrieval mainly comprises of four major steps, they are:

1. Defining image descriptor
2. Indexing data set
3. Similarity measurement
4. Searching

Defining Image Descriptor

One of our objective is to increase the retrieval efficiency by taking in consideration of colour, shape and texture combined features of the image. So here we have defined many features such as, colour: RGB histogram, HSV histogram; shape: fourier descriptor; texture: gray and hue LBP histogram, gray and hue CLBP histogram. For image descriptors, different sets of combinations of features(e.g. RGB histogram+fourier+gray LBP histogram) are concatenated linearly for the purpose of comparing the retrieval performance. Individual features(e.g.gray CLBP histogram) are also defined as image descriptor for comparing retrieval performance. Totally 12 combination of features are considered as image descriptor.

Indexing Data Set

The process of extracting features and storing them on persistent storage is commonly called “indexing”. Now we have defined our image descriptor, so in this stage will apply these image descriptor to all the images in the database individually

and the extracted features are stored as row array consequently in .CSV files. An array of (300,) is obtained as we used 60 images per class. Here we have obtained 7 .CSV files each consists of different features. They are linearly concatenated as per the needs, and used as the image descriptors to calculate similarity measurements using various distance metrics.

Similarity Measurement

The similarity measure is a significant task for retrieval of images from the database that is similar to query image. Retrieval performance of retrieval system also depends on the similarity measurement of the database image feature vector and query image feature vector [8]. Once the features are extracted from the indexed images, then retrieval becomes the measurement of similarity between the features. Many similarity measurements are exist. We have used Canberra distance, Euclidean distance, Manhattan distance and Chebyshev distances. Distances between query feature vector and features in database is computed. The images which have least distances are retrieved.

Canberra Distance:

It examines the sum of series of function of differences between coordinates of a pair of objects. Canberra distance [20] is calculated as below:

$$D = \sum_{i=1}^{n} \frac{|X_i - Y_i|}{|X_i| + |Y_i|} \tag{3.27}$$

Manhattan Distance:

The distance between two points measured along axes at right angles. In a plane with P1 at (X1, Y1) and P2 at (X2, Y2) it is,

$$D = |X_1 - X_2| + |Y_1 - Y_2| \tag{3.28}$$

Chebyshev Distance:

Chebyshev distance is a metric defined on a vector space where the distance between two vectors is the greatest of their differences along any coordinate dimension [21]. The Chebyshev distance between two vectors or points X and Y, with standard coordinates X_i and Y_i, respectively, is

$$D_{Chebyshev}(X, Y) := max(|X_i - Y_i|) \tag{3.29}$$

Mathematically, the Chebyshev distance is a metric induced by the supremum norm or uniform norm. In two dimensions, i.e. plane geometry, if the points P and Q have Cartesian (X_1, Y_1) and (X_2, Y_2) , their Chebyshev distance is

$$D_{Chebyshev}(X,Y) = max(|X_2 - X_1|, |Y_2 - Y_1|) \tag{3.30}$$

Euclidean Distance:
The Euclidean distance between two points in either the plane or 3-dimensional space measures the length of a segment connecting the two points.

$$D(X,Y) = \sqrt{\sum_{i=1}^{n} (X_i - Y_i)^2} \tag{3.31}$$

Image retrieval is done using all these similarity measurements and their performances will be compared. Highest accuracy providing similarity measurements will be used for the image retrieval for the final output.

Searching

Based on the input image given, the distances are computed and ranked. Top N images are retrieved and displayed where N is the threshold defined by user.

CHAPTER 4

RESULTS

4.1 Implementation

The above mentioned methods were implemented using **Python 3.6** as the main programming language and used **OpenCV , Numpy , Sklearn** libraries for image processing and classification. Visualization of results were carried out with the help of **Matplotlib**.

4.2 Classification

The model was trained with different kernels and hyperparameters. The following table shows the accuracy obtained in each case.

Kernel	C	gamma	degree	Accuracy
rbf	1	0.001	-	0.938
rbf	1	0.0001	-	0.919
rbf	10	0.001	-	0.938
rbf	10	0.0001	-	0.970
rbf	100	0.001	-	0.938
rbf	100	0.0001	-	0.970
rbf	1000	0.001	-	0.938
rbf	1000	0.0001	-	0.970
polynomial	1	-	1	0.957
polynomial	1	-	2	0.781
polynomial	1	-	3	0.710
polynomial	10	-	1	0.970
polynomial	10	-	2	0.900
polynomial	10	-	3	0.829
polynomial	100	-	1	0.970
polynomial	100	-	2	0.910
polynomial	100	-	3	0.876
polynomial	1000	-	1	0.970
polynomial	1000	-	2	0.919
polynomial	1000	-	3	0.881
linear	**1**	**-**	**-**	**0.97**1
linear	10	-	-	0.971
linear	100	-	-	0.971
linear	1000	-	-	0.971

Table 4.1: Hyperparamter Tuning

From the table4.1 we found the highest accuracy to be obtained by using linear kernel with C value as 1. We can conclude that, the features obtained are linearly separable by great extent.

The linear kernel SVC model is used for further classification problems. 5 - fold validated accuracy for various combinations of features was done to find the best possible combination.

Features	**5 - fold validated accuracy**
BGR hist	0.904
HSV hist	0.942
FD	0.608
Gray_LBP	0.880
Hue_LBP	0.897
Gray_CLBP	0.928
Hue_CLBP	0.917
BGR hist + FD + Gray_LBP	0.924
HSV hist + FD + Hue_LBP	0.957
BGR hist + FD + Gray_CLBP	0.964
Hue hist + FD + Hue_CLBP	**0.973**
BGR hist + HSV hist + FD + Gray_CLBP + Hue_CLBP	0.971

Table 4.2: Validated accuracy

The particular combination of HSV histogram, Fourier descriptors and CLBP of hue component in HSV colour space is found to give the best possible results. This is may be due to the fact HSV colour space considers actual colour component and invariant of illumination differences. Also completed local binary patterns are better than normal local binary patterns because of the extra information provided by magnitude differences between central and neighbouring pixels.

Confusion matrix was plotted for a 70:30 train test split of 300 images in database using linear SVC. 4.1 shows the Confusion matrix without normalization obtained from linear SVC kernel.

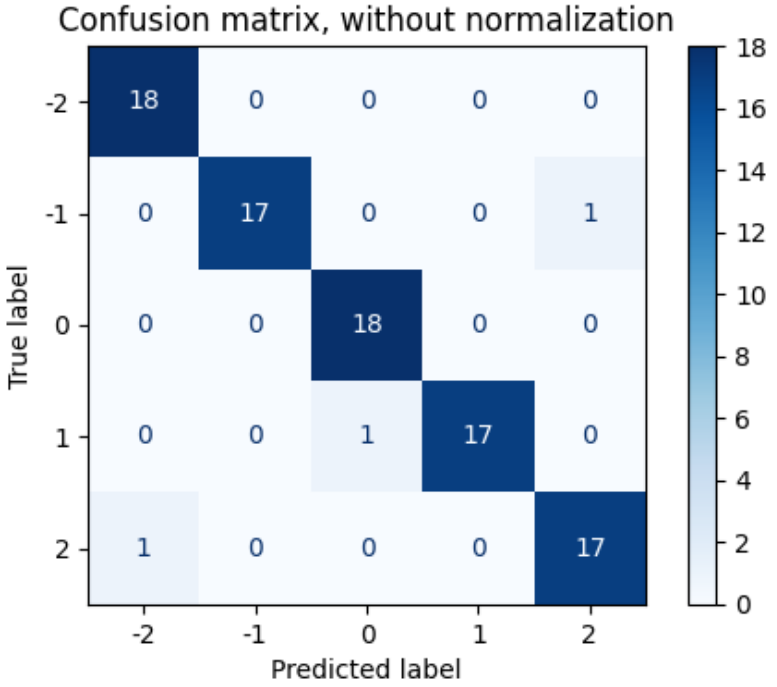


Figure 4.1: Confusion Matrix without Normalization

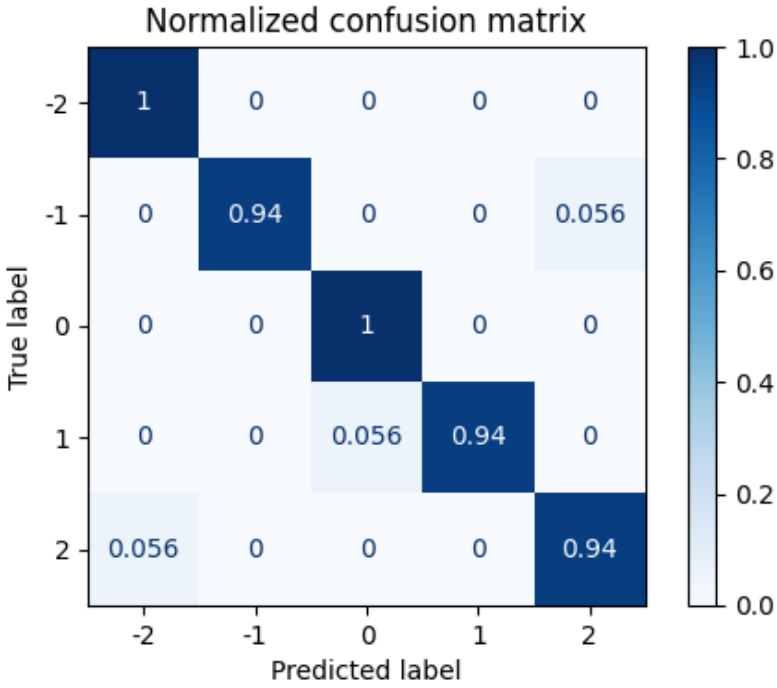


Figure 4.2: Normalized Confusion Matrix

Normalized Confusion matrix 4.2 is obtained by dividing each cell by support value ie., 18 images per class.

Precision, recall and F1 score was computed from confusion matrix and is mentioned in 4.3

Class	**Precision**	**Recall**	**F1 score**	**Support**
Bacterial Spot	0.947	1.000	0.973	18
Healthy	1.000	0.944	0.971	18
Late Blight	0.947	1.000	0.973	18
Late Blight	0.947	1.000	0.973	18
Septorial Spot	1.000	0.944	0.971	18
Mosaic	0.944	0.944	0.914	18
Macro Average	0.968	0.967	0.967	90

Table 4.3: Classification Metrics

The K - fold cross validated accuracy is computed and individual accuracy at each fold is plotted in 4.4

Fold	Accuracy
1	0.977
2	0.977
3	0.977
4	0.977
5	0.977
Mean	0.973

Table 4.4: 5- Fold Cross Validated Accuracy

The following images shows how accuracy of each class is varied with respect to change in number of training images.

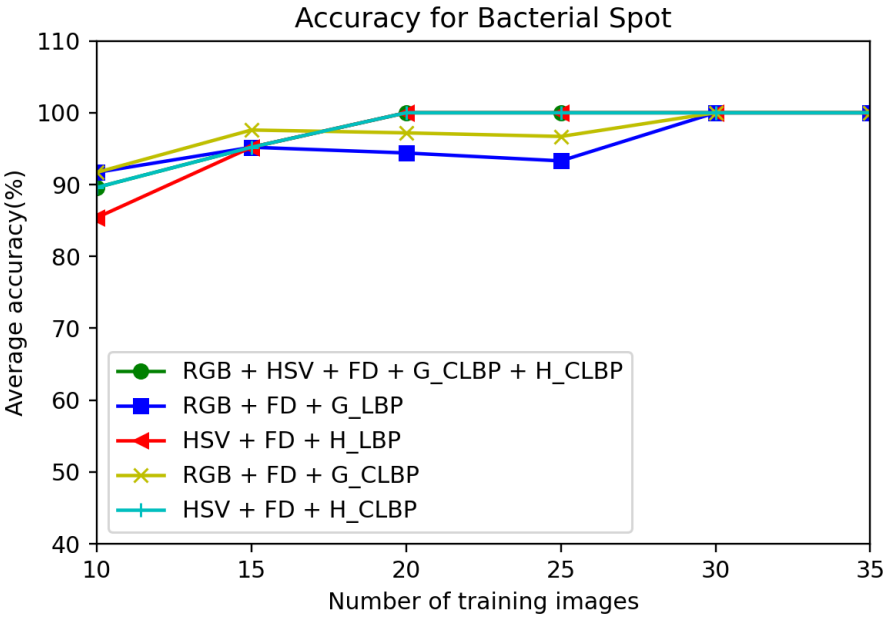


Figure 4.3: Bacterial Spot

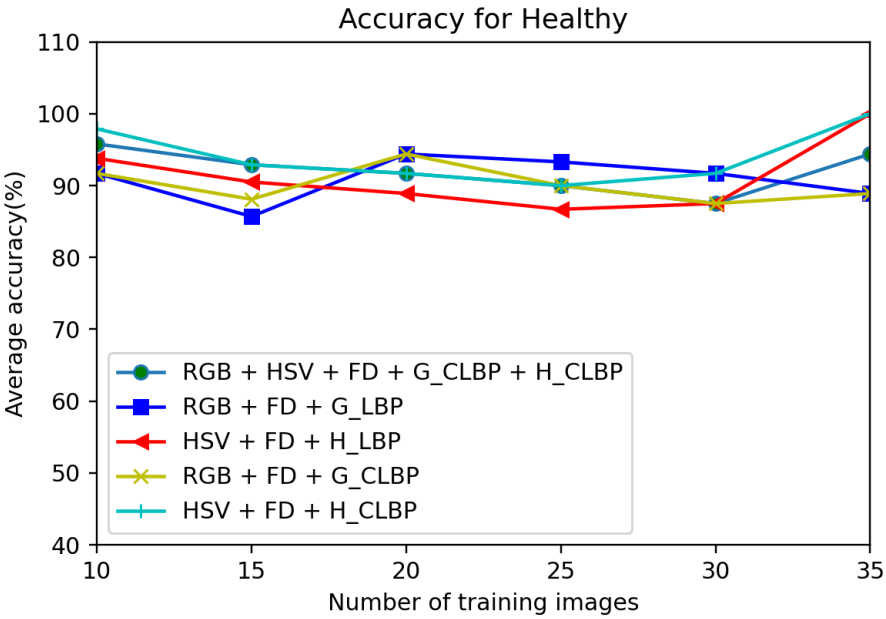


Figure 4.4: Healthy

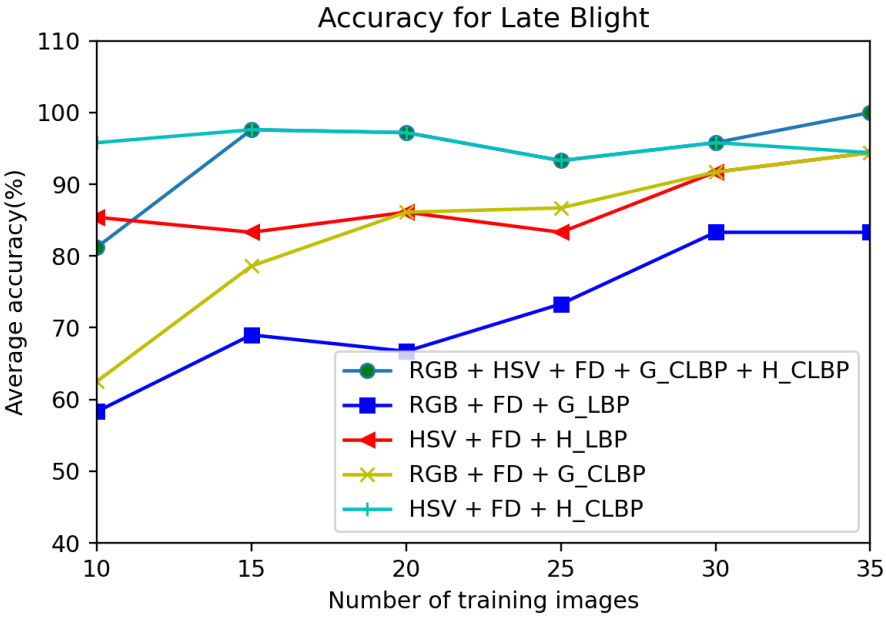


Figure 4.5: Late blight

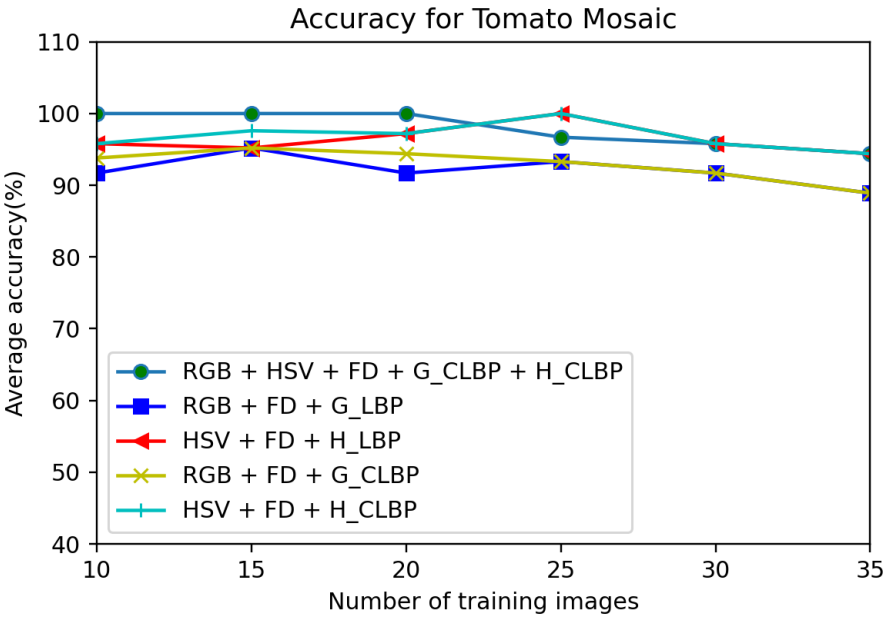


Figure 4.6: Mosaic

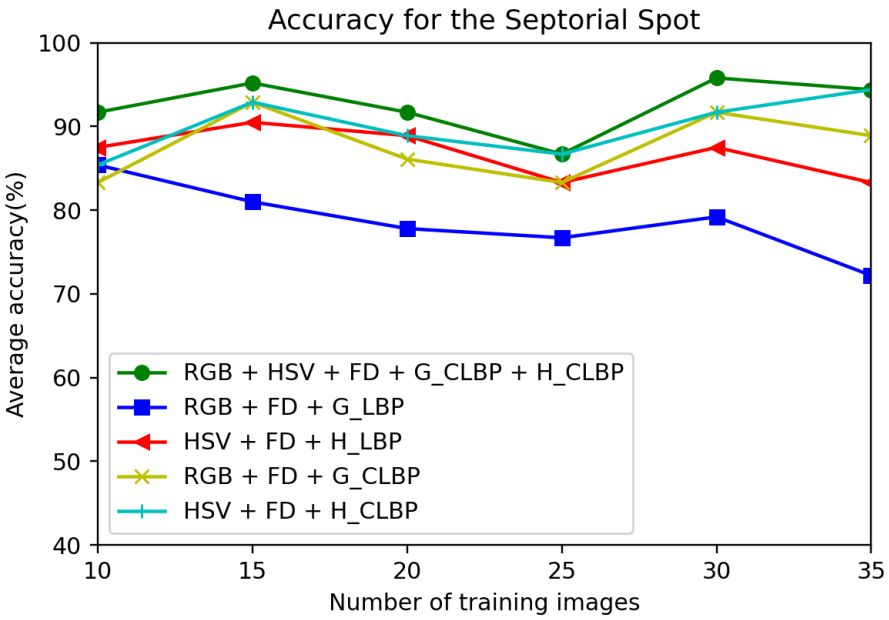


Figure 4.7: Septorial Spot

For each class the training accuracy increases with increase in images used for training. This can be attributed to the fact that additional distinguishable features are discovered from new images. Also the particular combination of HSV histogram, Fourier Descriptor and Complete LBP feature vectors is found to perform better for classification.

4.3 Retrieval

Content based retrieval system was developed based on different distance metrics. Top 5 relevant images are retrieved for each query image.

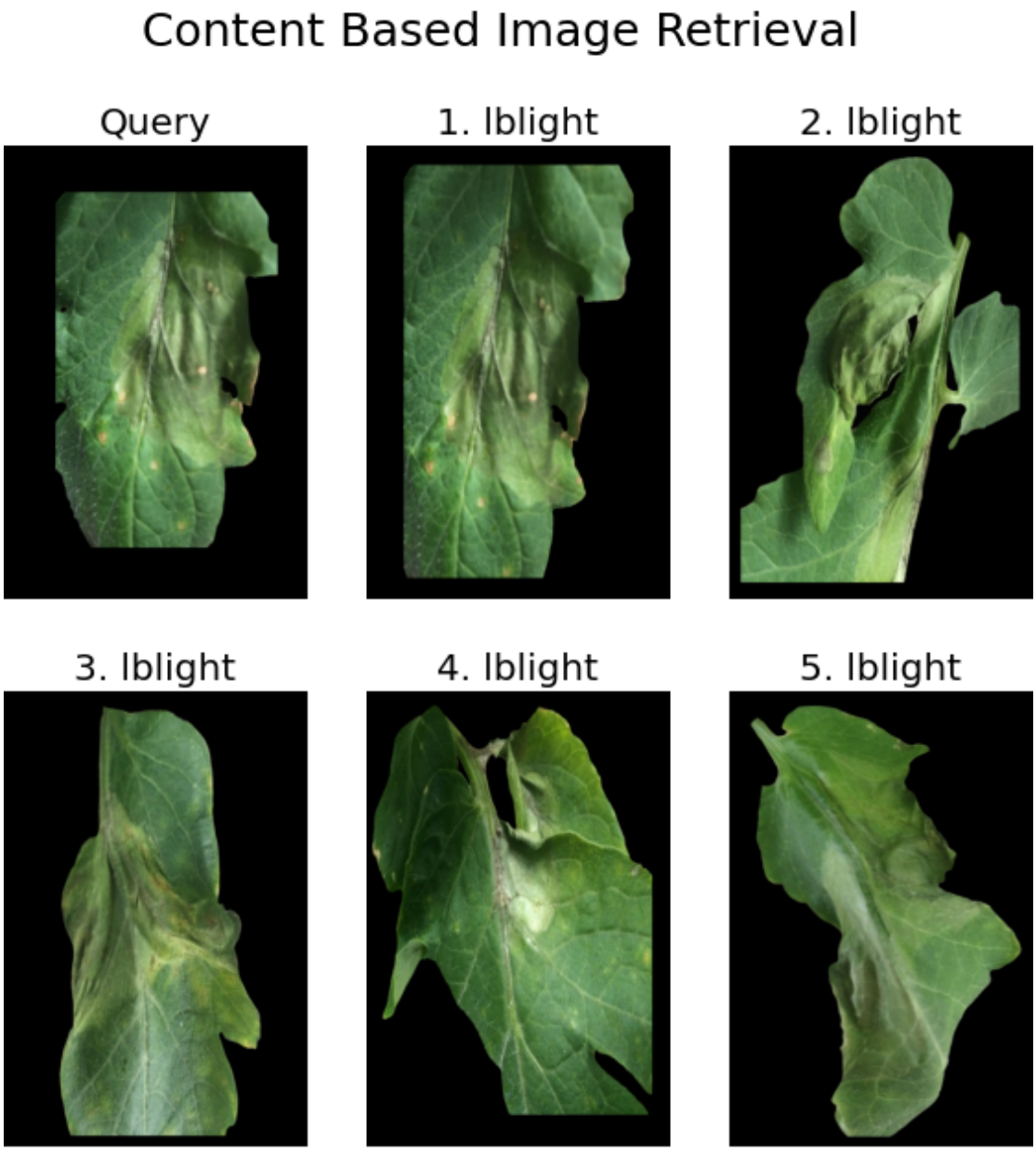


Figure 4.8: Retrieval output for Late Blight query image

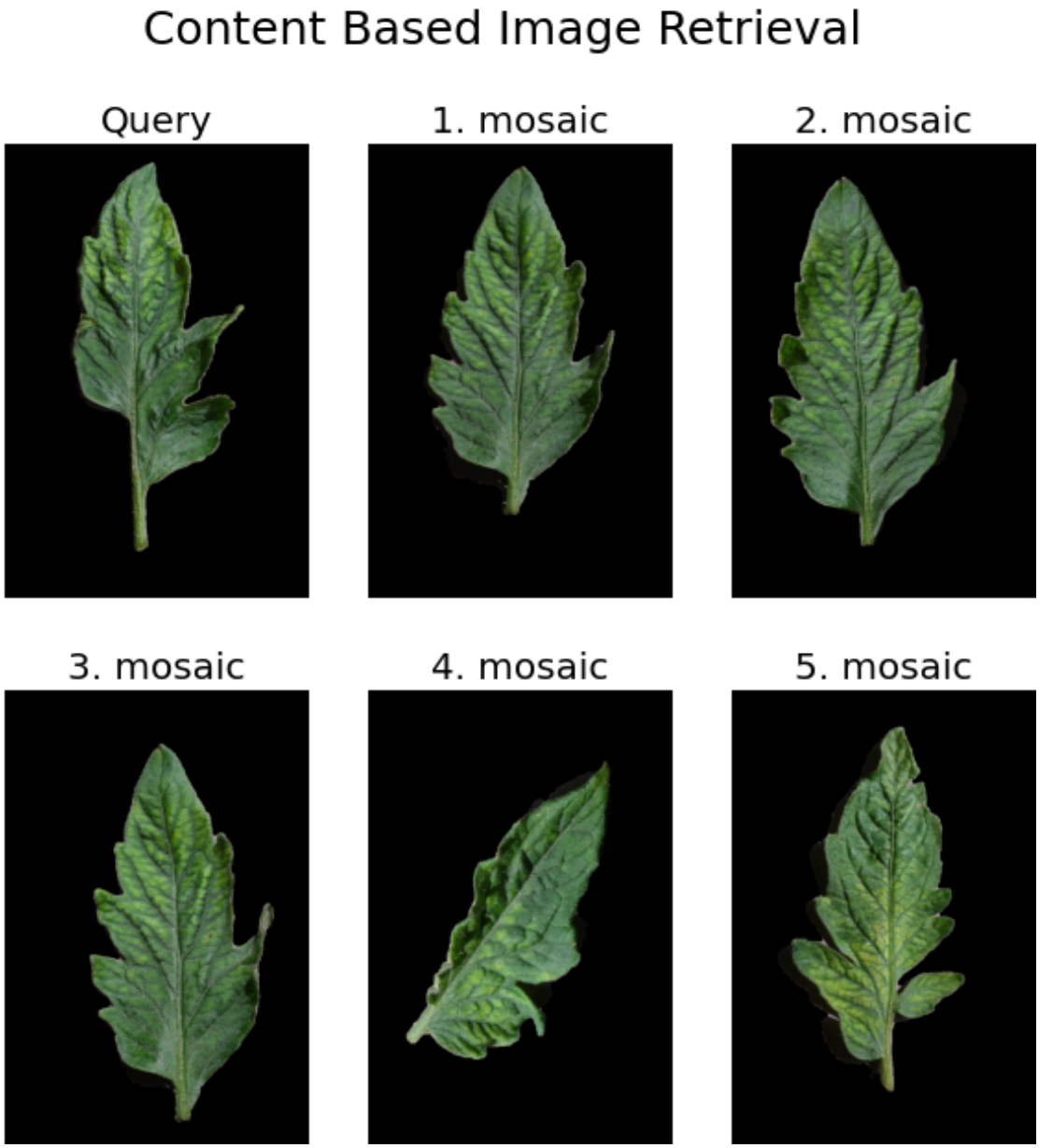


Figure 4.9: Retrieval output for Mosaic query image

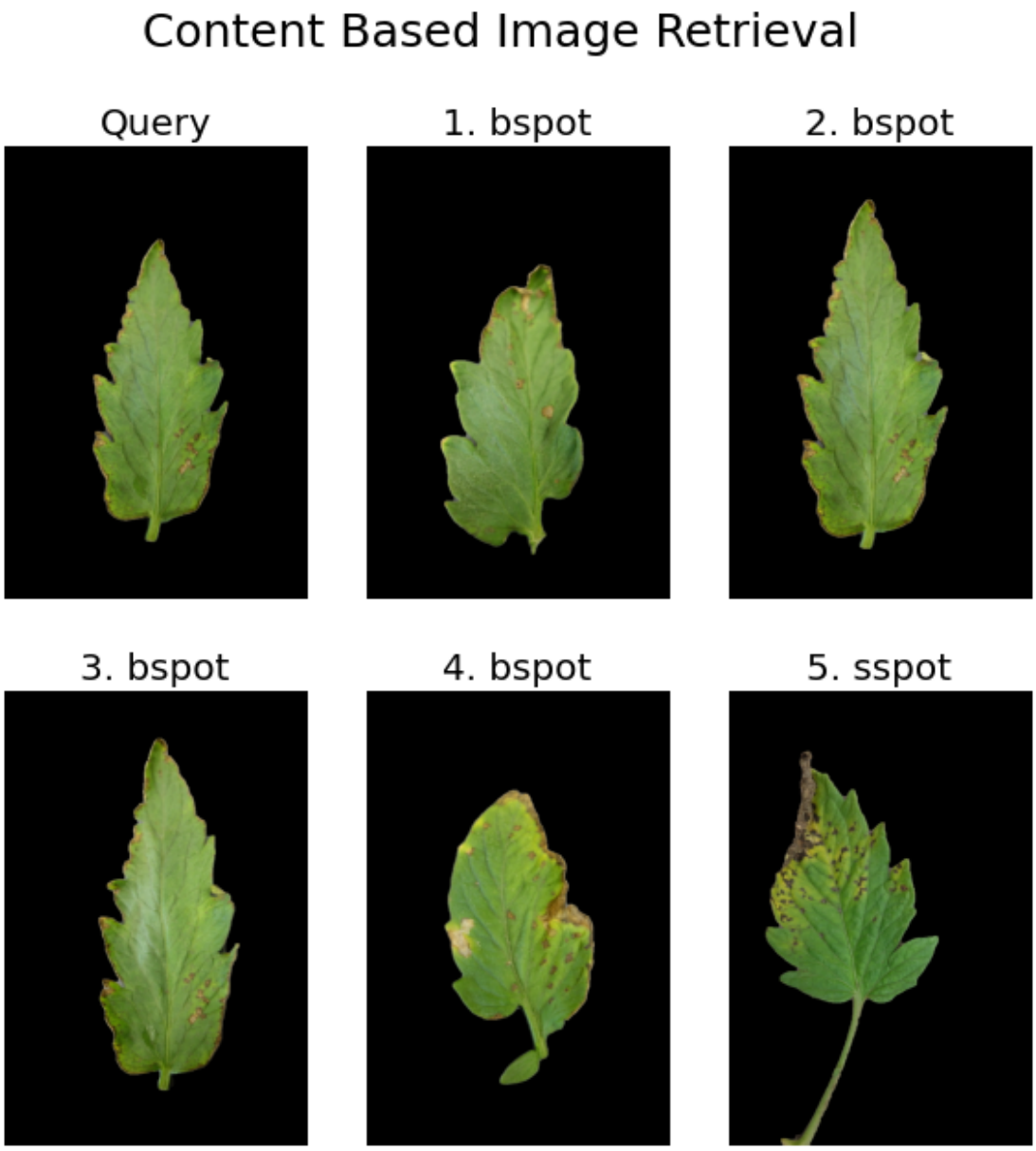


Figure 4.10: Retrieval output for Bacterial Spot query image

Retrieval accuracy is higher, but sometimes irrelevant images from other classes are also retrieved. But this is the case only when the number of images to be retrieved is increased.

Other performance metrics like Mean Average Precision and Bull's Eye Precision for different distance metrics are shown in 4.5 and 4.6 respectively.

Features	Distance Metric			
	Canberra	**Chebyshev**	**Euclidean**	**Manhattan**
BGR hist	0.7041	0.5356	0.5865	0.6129
HSV hist	0.8312	0.532	0.62196	0.6836
FD	0.341	0.3233	0.3567	0.3833
Gray_LBP	0.6562	0.4623	0.5053	0.5914
Hue_LBP	0.7287	0.3672	0.415	0.5376
Gray_CLBP	0.594	0.3706	0.4177	0.5141
Hue_CLBP	0.6167	0.3645	0.4069	0.5138
BGR hist + FD + Gray_LBP	0.7149	0.3233	0.3682	0.5591
HSV hist + FD + Hue_LBP	0.831	0.3237	0.3897	0.596
BGR hist + FD + Gray_CLBP	0.7394	0.3242	0.3788	0.5809
Hue hist + FD + Hue_CLBP	**0.8508**	0.3237	0.3898	0.5968
BGR hist + HSV hist + FD + Gray_CLBP + Hue_CLBP	0.8105	0.3241	0.4191	0.65

Table 4.5: Mean Average Precision

Features	Distance Metrics			
	Canberra	**Chebyshev**	**Euclidean**	**Manhattan**
BGR hist	0.7712	0.6245	0.6647	0.6909
HSV hist	0.8849	0.6437	0.674	0.744
FD	0.4725	0.5055	0.5159	0.5232
Gray_LBP	0.7192	0.5764	0.6062	0.6828
Hue_LBP	0.8131	0.4951	0.5172	0.5875
Gray_CLBP	0.6918	0.529	0.5497	0.6263
Hue_CLBP	0.6509	0.4951	0.5167	0.5782
BGR hist + FD + Gray_LBP	0.7794	0.5055	0.5193	0.6266
HSV hist + FD + Hue_LBP	0.884	0.5055	0.5232	0.6525
BGR hist + FD + Gray_CLBP	0.8003	0.5055	0.5218	0.6402
Hue hist + FD + Hue_CLBP	**0.9017**	0.5055	0.5232	0.6534
BGR hist + HSV hist + FD + Gray_CLBP + Hue_CLBP	0.867	0.5055	0.5289	0.6711

Table 4.6: Bull's Eye Performance

It is evident from the tables that highest performance is obtained by using Canberra distance on a combination of HSV histogram , Fourier Descriptors and Complete Local Binary Patterns on Hue colour space.

Precision - Recall curve is plotted for a particular query image as shown in 4.11

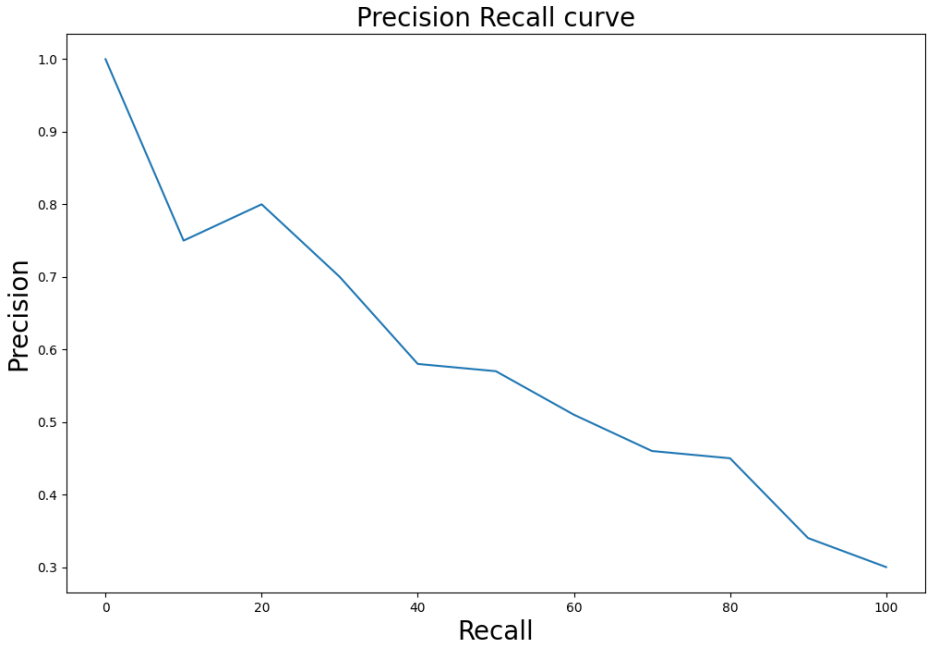


Figure 4.11: Precision Recall Curve

Precision recall curve should be a downward sloping curve. Since as number of images to be retrieved is increased, there are increasing chances of irrelevant images to be retrieved as well. Though this increases recall, precision decreases as a trade-off.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 Conclusions

Proposed CBIR system is particularly implemented for agricultural uses. Four tomato leaves diseases are taken into account. K- means clustering and bilateral filtering are done to clearly the segment the region of interest and remove the noise. The individual performance of colour, shape and texture features is found out. It is analyzed that if the features are fused, there is considerable increase in the result. The training and classification model is implemented using Support Vector Machines. The maximum mean accuracy of 97.3% is obtained for using combination of HSV histogram, Fourier descriptors and Uniform hue CLBP features, in classification. In regards of retrieval, the same combination of features provides a maximum mean precision of 85.08% and bull's eye perofrmance of 90.17%, using canberra distance metric. Though there are possibilities in improving, the proposed system offers impressive image retrieval performance.

5.2 Future Scope of Work

The scope of the research statement is enormous since everybody now has a handheld smartphone camera in their hand and necessary compute power inside it. So we could create a mobile application which scans the leaf and identification and classification of disease can be done. In addition to this parameters like percentage of disease affected, prevention and treatment methods are also can be implemented. Along with classification and identification, image search engine can be implemented. Further, customer feedback can be used as an input for the training model to improve the accuracy further up.

The result obtained provide a proof of concept that the methodologies discussed in the thesis can be used extensively for identification and classification of various other plants diseases. We have taken up only 4 tomato leaf diseases, the same can be explored further for other tomato leaf diseases too. Another area which we can improve is replacing deep learning algorithms instead of similarity measurements metrics for the content based image retrieval.

APPENDIX A

CODE ATTACHMENTS

A.1 Pre-processing

A.1.1 Resize images

```
import cv2
def resize(path):

    dim = (533,800)

    img = cv2.imread(path)
    resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
    return resized
```

A.1.2 Segmentation

```
import numpy as np
import cv2

def image_extractor(img):
    Z = img_hsv.reshape((-1,3))
        Z = np.float32(Z)
        criteria = (cv2.TERM_CRITERIA_EPS + cv2.
            TERM_CRITERIA_MAX_ITER,10,1.0)
        K = 2
        ret,label, center = cv2.kmeans(Z,K,None,criteria,10,cv2.
            KMEANS_RANDOM_CENTERS)
        center = np.uint8(center)
        res  = center[label.flatten()]
        seg = res.reshape((img_hsv.shape))
        # seg contains segmented image

        LOWER_GREEN = np.array([0,10,10])
        UPPER_GREEN = np.array([50,255,255])

        masked = cv2.inRange(seg,LOWER_GREEN,UPPER_GREEN)
        kernel = np.ones((7,7),np.uint8)
        ret, thresh = cv2.threshold(masked,110,255,cv2.THRESH_BINARY)
        opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel)
        closing = cv2.morphologyEx(opening,cv2.MORPH_CLOSE,kernel)

        _,contours,hierarchy = cv2.findContours(closing,cv2.
            RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
        ab = 0
        if len(contours) >1 :
                for index in range(len(contours)):
                        maxarea = cv2.contourArea(contours[0])
```

```
29                          if cv2.contourArea(contours[index]) > maxarea
                                :
30                                  maxarea = cv2.contourArea(contours[
                                        index])
31                                  ab = index
32
33          black = np.zeros((height,width),np.uint8)
34          cv2.drawContours(black,contours,ab,(255,255,255),-1)
35          blur = cv2.GaussianBlur(black,(7,7),0)
36
37          ret, binary = cv2.threshold(blur,127,255,0)
38
39          extract = cv2.bitwise_and(img,img, mask= thresh2)
40      # extract contains leaf ROI over black background
41      # binary contains binary image
42          return binary,extract
```

A.2 Feature Extraction

A.2.1 Colour histogram

```
1
2  import numpy as np
3  import cv2
4
5  read_path='image_path'
6
7  bgr_img = cv2.imread(read_path)
8  hist_b = np.asarray(cv2.calcHist([bgr_img],[0],None,[255],[1,255]))
9  hist_g = np.asarray(cv2.calcHist([bgr_img],[1],None,[255],[1,255]))
10 hist_r = np.asarray(cv2.calcHist([bgr_img],[2],None,[255],[1,255]))
11 hist_b = hist_b.T[0]
12 hist_g = hist_g.T[0]
13 hist_r = hist_r.T[0]
14 hist = np.concatenate((hist_b,hist_g))
15 hist = np.concatenate((hist,hist_r))
16 X.append(hist)
17 features = np.asarray(X)
```

A.2.2 Fourier Descriptors

```
1  import cv2
2  import numpy as np
3
4
5  MIN_DESCRIPTOR = 30
6
7  def findDescriptor(img):
8          contour = []
9          _,contour, hierarchy = cv2.findContours(img,cv2.RETR_EXTERNAL
               ,cv2.CHAIN_APPROX_NONE)
10         contour_array = contour[0][:,0,:]
11         contour_complex = np.empty(contour_array.shape[:-1],dtype =
               complex)
12         contour_complex.real = contour_array[:, 0]
```

```
13          contour_complex.imag = contour_array[:,-1]
14          fourier_result = np.fft.fft(contour_complex)
15          return fourier_result
16
17
18  def truncate_descriptor(descriptors, degree):
19      """this function truncates an unshifted fourier descriptor array
20      and returns one also unshifted"""
21      descriptors = np.fft.fftshift(descriptors)
22      center_index = len(descriptors) / 2
23      #center_index = int(center_index)
24
25      descriptors = descriptors[int(center_index - degree / 2):int(
            center_index + degree / 2)]
26      descriptors = np.fft.ifftshift(descriptors)
27      return descriptors
28
29
30  def reconstruct(descriptors, degree):
31      """ reconstruct(descriptors, degree) attempts to reconstruct the
            image
32      using the first [degree] descriptors of descriptors"""
33      # truncate the long list of descriptors to certain length
34      descriptor_in_use = truncate_descriptor(descriptors, degree)
35      contour_reconstruct = np.fft.ifft(descriptor_in_use)
36      contour_reconstruct = np.array(
37          [contour_reconstruct.real, contour_reconstruct.imag])
38      contour_reconstruct = np.transpose(contour_reconstruct)
39      contour_reconstruct = np.expand_dims(contour_reconstruct, axis=1)
40      # make positive
41      if contour_reconstruct.min() < 0:
42          contour_reconstruct -= contour_reconstruct.min()
43      # normalization
44      contour_reconstruct *= 800 / contour_reconstruct.max()
45      # type cast to int32
46      contour_reconstruct = contour_reconstruct.astype(np.int32, copy=
            False)
47      black = np.zeros((800, 800), np.uint8)
48      # draw and visualize
49      cv2.drawContours(black, contour_reconstruct, -1, 255, thickness
            =10)
50      return descriptor_in_use
```

A.2.3 Local Binary Pattern

```
1   import cv2
2   import os
3   import numpy as np
4   from matplotlib import pyplot as plt
5
6   def get_pixel(img, center, x, y):
7       new_value = 0
8       try:
9           if img[x][y] >= center:
10              new_value = 1
11      except:
```

```
12          pass
13      return new_value
14
15  def lbp_calculated_pixel(img, x, y):
16      '''
17       64 | 128 |   1
18      ----------------
19       32 |   0 |   2
20      ----------------
21       16 |   8 |   4
22      '''
23      center = img[x][y]
24      val_ar = []
25      val_ar.append(get_pixel(img, center, x-1, y+1))     # top_right
26      val_ar.append(get_pixel(img, center, x, y+1))       # right
27      val_ar.append(get_pixel(img, center, x+1, y+1))     #
            bottom_right
28      val_ar.append(get_pixel(img, center, x+1, y))       # bottom
29      val_ar.append(get_pixel(img, center, x+1, y-1))     # bottom_left
30      val_ar.append(get_pixel(img, center, x, y-1))       # left
31      val_ar.append(get_pixel(img, center, x-1, y-1))     # top_left
32      val_ar.append(get_pixel(img, center, x-1, y))       # top
33
34      power_val = [1, 2, 4, 8, 16, 32, 64, 128]
35      val = 0
36      for i in range(len(val_ar)):
37          val += val_ar[i] * power_val[i]
38      return val
39
40  read_path = 'image path'
41  img_bgr=cv2.imread(read_path)
42  height, width,channel= img_bgr.shape
43  #hsv_img = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)
44  img_gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
45  #img_h, img_s, img_v = hsv_img[:, :, 0], hsv_img[:, :, 1], #hsv_img
        [:, :, 2]
46  #img_h = cv2.bilateralFilter(img_h,11,17,17)
47  img_gray=cv2.bilateralFilter(img_gray,11,17,17)
48  img_lbp = np.zeros((height, width,3), np.uint8)
49  for i in range(0, height):
50    for j in range(0, width):
51      #img_lbp[i, j] = lbp_calculated_pixel(img_h, i, j)
52      img_lbp[i, j] = lbp_calculated_pixel(img_gray, i, j)
53
54  hist=cv2.calcHist([img_lbp],[0],None,[255],[1,255])
55  plt.plot(hist,color='gray',alpha = 0.7)
56  plt.xlim([0,256])
57  plt.xlabel('Bins')
58  plt.ylabel('Frequency')
59  plt.show()
```

A.2.4 Complete Local Binary Pattern and Non Rotational Uniform LBP

```
1  import cv2
2  import os
3  import numpy as np
```

```
4  from matplotlib import pyplot as plt
5
6  powarr = [64,128,1 ,32,0, 2, 16, 8, 4 ]
7  temp = np.zeros(9)
8
9  def nri_uniform():
10         a = [  1, 3, 7, 15, 31, 63, 127 ]
11         b = []
12         INT_BITS = 8
13         d=1
14
15
16         for j in range(len(a)):
17                 for i in range(0,8):
18                         b.append(((a[j] << i)|(a[j] >> (INT_BITS - i)
                               ))&0b11111111)
19                         #b = np.unique(b)
20         b.append(0)
21         b.append(255)
22         b.sort()
23         return b
24
25 def get_val_nri_uniform(n):
26         if n in b :
27                 return b.index(n)
28         else:
29                 return 59
30
31 def pixel(img,x,y):
32
33     tem = img[x-1:x+2,y-1:y+2]
34
35     a = [-1,0,1]
36     k =0
37     for i in a:
38       for j in a:
39         temp[k] = tem[i,j]
40         k+= 1
41     cent = temp[4]
42     sval = 0
43     mval = 0
44     cval = 0
45     if img[x][y]>=c:
46       cval=255
47     for z in range(0,9):
48       if temp[z]>=cent:
49         sval+= powarr[z]
50       if abs(int(temp[z])-int(cent)) > c:
51         mval+= powarr[z]
52
53     return sval,mval,cval
54
55 read_path = 'image_path'
56 X = []
57 count=1
58 b = nri_uniform()
```

```
59
60
61 img=cv2.imread(read_path,0)
62 #img=cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
63 img = cv2.bilateralFilter(img,11,17,17)
64 #cv2_imshow(img)
65 height,width= img.shape
66 s = np.zeros((height, width), np.uint8)
67 m = np.zeros((height, width), np.uint8)
68 ce = np.zeros((height, width), np.uint8)
69 nri_s = np.zeros((height, width), np.uint8)
70 nri_m = np.zeros((height, width), np.uint8)
71 #img,s,v=img[:,:,0],img[:,:,1],img[:,:,2]
72 #print(h.shape)
73 c = int(np.mean(img))
74 print(count, c)
75 count+=1
76 for i in range(1, height-1):
77     for j in range(1, width-1):
78         s[i,j],m[i,j],ce[i,j] = pixel(img,i,j)
79         nri_s[i,j] = get_val_nri_uniform(s[i][j])
80         nri_m[i,j] = get_val_nri_uniform(m[i][j])
81 X.append(s.flatten())
82 d_clbp=np.concatenate((s,m,ce))
83 final = np.concatenate((nri_s,nri_m))
84
85 hist_gray_s=cv2.calcHist([s],[0],None,[255],[1,255])
86 hist_gray_m=cv2.calcHist([m],[0],None,[255],[1,255])
87 hist_gray_nris=cv2.calcHist([nri_s],[0],None,[59],[1,59])
88 hist_gray_nrim=cv2.calcHist([nri_m],[0],None,[59],[1,59])
89
90 plt.plot(hist_gray_s,color = 'b',alpha = 0.7)
91 #plt.xlim([1,256])
92 plt.xlabel('Bins')
93 plt.ylabel('Frequency')
94 plt.show()
95
96 plt.plot(hist_gray_m,color='g',alpha=0.7)
97 #plt.xlim([1,256])
98 plt.xlabel('Bins')
99 plt.ylabel('Frequency')
100 plt.show()
101
102 plt.plot(hist_gray_nris,color='b',alpha=0.7)
103 #plt.xlim([0,60])
104 plt.xlabel('Bins')
105 plt.ylabel('Frequency')
106 plt.show()
107
108 plt.plot(hist_gray_nrim,color='g',alpha=0.7)
109 #plt.xlim([0,60])
110 plt.xlabel('Bins')
111 plt.ylabel('Frequency')
112 plt.show()
```

A.3 Classification

```
import numpy as np
import os
from numpy import savetxt,loadtxt

################## Files  ############################
# 1: bgr hist
# 2: hsv hist
# 3: shape
# 4: gray_clbp_S_M
# 5: hue_clbp_S_M
# 6: gray_LBP
# 7: hue_LBP

################## Constants ######################
B_SPOTS = -2
HEALTHY = -1
L_BLIGHT = 0
T_MOSAIC = 1
S_SPOTS = 2
diseases = ['bspot','healthy','lblight','mosaic','sspot']
label_dict = {'bspot':B_SPOTS,'healthy':HEALTHY,'lblight':L_BLIGHT,'
    mosaic':T_MOSAIC,'sspot':S_SPOTS}
files = [1,2,3,4,5]


################ Variables ##########################
X = []
Y = []
y = []




################ Feature Vector ####################
path = 'drive/My_Drive/Features'
flag = 0
flag2 = 0
for disease in diseases:
  read_path = os.path.join(path,disease)
  for file in files:
    f_name = str(file)+'.csv'
    file_path = os.path.join(read_path,f_name)

    if flag == 0:
      data = loadtxt(file_path,delimiter = ',')
      data = np.asarray(data)

    if len(files) >=2 and flag == 1:
      data1 = loadtxt(file_path,delimiter = ',')
      data1 = np.asarray(data1)
      data = np.concatenate((data,data1),axis = 1)


```

```
53
54        flag = 1
55      print(data.shape)
56      flag = 0
57      temp = np.zeros(60)
58      temp.fill(label_dict[disease])
59      if flag2 == 0:
60        feature_data = data
61        label = temp
62      if flag2 == 1:
63        feature_data = np.concatenate((feature_data,data))
64        label = np.concatenate((label,temp))
65      flag2 = 1
66
67  features = feature_data
68  print(feature_data.shape)
69  print(label.shape)
70
71  ###################### Feature Scaling ########################
72
73  from sklearn.preprocessing import StandardScaler
74  from sklearn.model_selection import train_test_split
75  X_train, X_test, y_train, y_test = train_test_split(features, label,
        test_size=0.3, random_state=20,stratify = label)
76
77  scaler = StandardScaler()
78  X_train = scaler.fit_transform(X_train)
79  X_test = scaler.fit_transform(X_test)
80
81  ################### Hyperparamter Tuning ######################
82  from sklearn import svm
83  from sklearn.model_selection import GridSearchCV
84
85  parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
86                       'C': [1, 10, 100, 1000]},
87                      {'kernel': ['poly'], 'C': [ 1,10, 100, 1000,],'
                           degree':[1,2,3]},
88                {'kernel': ['linear'], 'C': [ 1,10, 100, 1000,]} ]
89  svc = svm.SVC()
90  clf = GridSearchCV(svc,parameters,scoring='accuracy')
91  clf.fit(X_train,y_train)
92  print(clf.best_score_)
93  print(clf.best_params_)
94
95  means = clf.cv_results_['mean_test_score']
96  stds = clf.cv_results_['std_test_score']
97  for mean, std, params in zip(means, stds, clf.cv_results_['params']):
98    print("%0.3f (+/-%0.03f) for %r"% (mean, std * 2, params))
99
100
101 ################## K Fold Validated Accuracy ##################
102 from sklearn.model_selection import cross_val_score,
        StratifiedShuffleSplit,cross_validate
103 from sklearn.pipeline import make_pipeline
104
105 cv = StratifiedShuffleSplit(n_splits = 5, test_size = 0.3,
```

```
    random_state = 2)

classifier = svm.SVC(kernel = 'linear')

clf = make_pipeline(StandardScaler(),classifier)
score  = cross_val_score(clf,features,label,cv = cv)
print(score)
print(score.mean())

cv_results = cross_validate(clf, features, label,cv = 5 )
print(cv_results['test_score'])
print(cv_results['test_score'].mean())

################ Confusion Matrix  ############################

import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix
from matplotlib.pyplot import figure

np.set_printoptions(precision=2)
class_names = ['B_SPOTS','HEALTHY','L_BLIGHT','T_MOSAIC','S_SPOTS']
classifier.fit(X_train,y_train)
y_pred = classifier.predict(X_test)
y_true = y_test
# Plot non-normalized confusion matrix
titles_options = [("Confusion matrix, without normalization", None),
                  ("Normalized confusion matrix", 'true')]
label = [-2,-1,0,1,2]
for title, normalize in titles_options:
    disp = plot_confusion_matrix(classifier, X_test, y_test,
                                 display_labels=label,
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
    disp.ax_.set_title(title)
    # print(title)
    # print(disp.confusion_matrix)


fig = plt.gcf()
plt.show()
#fig.savefig('conf_matrix_norm2.png',dpi = 100)
print(classification_report(y_true, y_pred, digits=3))
```

A.4 Retrieval

```
import numpy as np
from sklearn import svm
from sklearn.utils import shuffle
import os
from numpy import savetxt,loadtxt

################# Constants ######################
B_SPOTS = -2
HEALTHY = -1
L_BLIGHT = 0
```

```
T_MOSAIC = 1
S_SPOTS = 2
diseases = ['bspot','healthy','lblight','mosaic','sspot']
label_dict = {'bspot':B_SPOTS,'healthy':HEALTHY,'lblight':L_BLIGHT,'
    mosaic':T_MOSAIC,'sspot':S_SPOTS}
files = [1,2,3,4,5]
################ Variables ##########################
X = []
Y = []
y = []


path = 'drive/My Drive/Features/'
flag = 0
flag2 = 0
for disease in diseases:
  read_path = os.path.join(path,disease)
  for file in files:
    f_name = str(file)+'.csv'
    file_path = os.path.join(read_path,f_name)

    if flag == 0:
      data = loadtxt(file_path,delimiter = ',')
      data = np.asarray(data)

    if len(files) >=2 and flag == 1:
      data1 = loadtxt(file_path,delimiter = ',')
      data1 = np.asarray(data1)
      data = np.concatenate((data,data1),axis = 1)



    flag = 1
  print(data.shape)
  flag = 0
  temp = np.zeros(60)
  temp.fill(label_dict[disease])
  if flag2 == 0:
    feature_data = data
    label = temp
  if flag2 == 1:
    feature_data = np.concatenate((feature_data,data))
    label = np.concatenate((label,temp))
  flag2 = 1

features = feature_data
print(feature_data.shape)
print(label.shape)

########### Precison Recall Curves #################
data = features
from sklearn.neighbors import NearestNeighbors

low = [0,60,120,180,240]
up = [60,120,180,240,300]

```

```
nneigh = 200
neigh1 = NearestNeighbors(n_neighbors=nneigh, metric='canberra')
neigh1.fit(data)
N = 50
x = N//60
sc,ind = neigh1.kneighbors([data[N]])
ind = ind[0]
ind = ind[1:]
relevant = np.zeros(nneigh-1)
precision = np.zeros(nneigh-1)
recall =  np.zeros(nneigh-1)
#print(ind)
origc = 0
count = 0
for i in ind:
  if i >=low[x] and i < up[x]:
    count+=1
  relevant[origc] = count
  precision[origc] = count/(origc+1)
  recall[origc] = (count/59)*100
  origc+=1
print(count)
print(relevant)
print(precision)
print(recall)


import matplotlib.pyplot as plt

fnt = {'fontsize':20}
plt.figure(figsize=(12,8),dpi = 100)
#plt.plot(recall,precision)



plt.xlabel('Recall',fnt)
plt.ylabel('Precision',fnt)
plt.title('Precision Recall curve',fnt)
plt.savefig('pr curve disc.png')

############## Bull's Eye Performance ##########################

data = features
from sklearn.neighbors import NearestNeighbors

low = [0,60,120,180,240]
up = [60,120,180,240,300]

metrics = ['canberra','chebyshev','euclidean','manhattan']
for metr in metrics:
  nneigh = 119
  neigh1 = NearestNeighbors(n_neighbors=nneigh, metric=metr)
  neigh1.fit(data)

  sum = 0

```

```
122   for N in range(0,300):
123     x = N//60
124     sc,ind = neigh1.kneighbors([data[N]])
125     ind = ind[0]
126     ind = ind[1:]
127     #print(ind)
128     count = 0
129     for i in ind:
130       if i >=low[x] and i < up[x]:
131         count+=1
132     sum += count/59
133   print(sum/300)
134
135   ############### Mean Average Precision ############################
136 data = features
137 from sklearn.neighbors import NearestNeighbors
138
139 low = [0,60,120,180,240]
140 up = [60,120,180,240,300]
141
142
143 nneigh = 61
144 neigh1 = NearestNeighbors(n_neighbors=nneigh,metric='euclidean')
145 neigh1.fit(data)
146
147 relevant = np.zeros(nneigh)
148 avg_prec = 0
149 N = 4
150 for N in range(0,300):
151   x = N//60
152   sc,ind = neigh1.kneighbors([data[N]])
153   ind = ind[0]
154   ind = ind[1:]
155   count = 0
156   relc = 0
157   sum = 0
158   for i in ind:
159     if i >=low[x] and i < up[x]:
160       sum = sum+1
161       relc  =relc+1
162       relevant[count] = sum
163     else:
164       relevant[count] = sum
165     relevant[count] = relevant[count]/(count+1)
166     count = count+1
167   # print(relevant)
168   # print(relc)
169   #print(np.mean(relevant))
170   avg_prec = avg_prec + np.mean(relevant)
171 print(avg_prec/300)
172
173 ############################### CBIR ########################
174
175 data = features
176 from sklearn.neighbors import NearestNeighbors
177 nneigh = 6
```

```
neigh1 = NearestNeighbors(n_neighbors=nneigh, metric = 'canberra')
neigh1.fit(data)
N = 6
sc,ind = neigh1.kneighbors([data[N]])
ind = ind[0]
print(ind)
low = [0,60,120,180,240]
up = [60,120,180,240,300]
sc,ind = neigh1.kneighbors([data[N]])
ind = ind[0]
#print(ind)
count = 0
x = 0
for i in ind:
  fold_no = i//60 +1
  file_no = i%60 +1

  #print(fold_no,file_no)
  if i>=low[x] and i <up[x]:
    count = count+1

print(count)


fnt = {'fontsize':20}
columns = 3
rows = 2
fig=plt.figure(figsize=(10, 10))
print(ind)
for i in range(1, columns*rows +1):
    print(ind[i-1])
    database =  'drive/My Drive/Dataset/' + db[(ind[i-1]//60)]+'/
        Extracted Image/extract'+str(ind[i-1]%60 +1)+'.png'
    img = cv2.imread(database)
    img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    fig.add_subplot(rows, columns, i)
    if i == 1:
      plt.title("Query",fnt)
    else:
      title = str(i-1)+'. '+diseases[(ind[i-1]//60)]

      plt.title(title,fnt)
    plt.imshow(img)
    plt.axis('off')
plt.xticks([])
plt.yticks([])
fig.suptitle("Content Based Image Retrieval",fontsize = 25 )
plt.savefig('drive/My Drive/Results/imperfect.png')
plt.show()
```

REFERENCES

[1] Monika Jhuria, Ashwani Kumar, and Rushikesh Borse. “Image processing for smart farming: Detection of disease and fruit grading”. In: *2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013)* (2013), pp. 521–526.

[2] Sachin Dubey and Anand Singh Jalal. “Detection and Classification of Apple Fruit Diseases Using Complete Local Binary Patterns”. In: *2012 Third International Conference on Computer and Communication Technology* (2012), pp. 346–351.

[3] Sagar Vetal and R S Khule. “Tomato Plant Disease Detection using Image Processing”. In: *International Journal of Advanced Research in Computer and Communication Engineering* 6 (2017), pp. 293–297.

[4] Ch. Mamatha Y. Vijaya Bhaskar Reddy Dr. S. Sai Satyanarayana Reddy Priyadarshini Chatterjee. *USE OF IMAGE PROCESSING TECHNIQUES TO DETECT DISEASES IN TOMATO LEAVES*. URL: http://www.iaeme.com/IJCIET/issues.asp?JType=IJCIET.

[5] Jayamala Kumar Patil and Raj Kumar. “Analysis of content based image retrieval for plant leaf diseases using color, shape and texture features”. In: *Engineering in Agriculture, Environment and Food, Elsevier* 10 (Apr. 2017), pp. 69–78. DOI: 10.1016/j.eaef.2016.11.004.

[6] Shiv Ram Dubey and Anand Singh Jalal. “Apple disease classification using color, texture and shape features from images”. In: *Signal, Image and Video Processing, Springer* 10.5 (July 2016), pp. 819–826. DOI: 10.1007/s11760-015-0821-1.

[7] Ekta Walia Chandan Singh and Kanwal Preet Kaur. “Color texture description with novel local binary patterns for effective image retrieval”. In: *Pattern Recognition, Elsevier* 76 (Apr. 2018), pp. 50–68. DOI: 10.1016/j.patcog.2017.10.021.

[8] Diksha Kurchaniya and Punit Johari. “Analysis of Different Similarity Measures in Image Retrieval Based on Texture and Shape”. In: 4 (Mar. 2018).

[9] G. B. Coleman and H. C. Andrews. “Image segmentation by clustering”. In: *Proceedings of the IEEE* 67.5 (1979), pp. 773–785.

[10] Rishav Chakravarti and Xiannong Meng. “A Study of Color Histogram Based Image Retrieval”. In: *Sixth International Conference on Information Technology: New Generations* (May 2009), pp. 1323–1328. DOI: 10.1109/ITNG.2009.126.

[11] A. Folkers and H. Samet. “Content-based image retrieval using Fourier descriptors on a logo database”. In: *Object recognition supported by user interaction for service robots*. Vol. 3. 2002, 521–524 vol.3.

[12] T. Ojala, M. Pietikainen, and T. Maenpaa. “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7 (2002), pp. 971–987.

[13] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. “Face Recognition with Local Binary Patterns”. In: vol. 3021. May 2004, pp. 469–481. DOI: 10.1007/978-3-540-24670-1_36.

[14] S. Malathi and C. Meena. “FINGERPRINT MATCHING BASED ON PORE CENTROIDS”. In: *ICTACT Journal on Image and Video Processing* 01 (2011), pp. 220–223.

[15] Loris Nanni, Alessandra Lumini, and Sheryl Brahnam. “Local binary pattern variants as texture descriptors for medical image analysis”. In: *Artificial intelligence in medicine* 49 (Mar. 2010), pp. 117–25. DOI: 10.1016/j.artmed.2010.02.006.

[16] Z. Guo, L. Zhang, and D. Zhang. “A Completed Modeling of Local Binary Pattern Operator for Texture Classification”. In: *IEEE Transactions on Image Processing* 19.6 (2010), pp. 1657–1663.

[17] Tobias Lindahl. *Study of local binary patterns*. 2007.

[18] Remco Veltkamp and Mirela Tanase. “Content-Based Image Retrieval Systems: A Survey”. In: *Technical report, Utrecht University* (Nov. 2000).

[19] H. Tamura, S. Mori, and T. Yamawaki. “Textural Features Corresponding to Visual Perception”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 8.6 (1978), pp. 460–473.

[20] G. N. Lance and W. T. Williams. “A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems”. In: *The Computer Journal* 9.4 (Feb. 1967), pp. 373–380. ISSN: 0010-4620. DOI: 10.1093/comjnl/9.4.373. eprint: https://academic.oup.com/comjnl/article-pdf/9/4/373/1101470/9-4-373.pdf. URL: https://doi.org/10.1093/comjnl/9.4.373.

[21] Cyrus D Cantrell. *Modern mathematical methods for physicists and engineers*. Cambridge University Press, 2000.