Leaf Classification using Convolutional Neural Network

A thesis

submitted in partial fulfillment of the requirement for the Degree of

Master of Computer Science and Engineering

of

Jadavpur University

By

Sayantan Maity

Registration No.: 140763 of 2017-2018

Examination Roll No.: M4CSE19008

Under the Guidance of

Prof. Susmita Ghosh

Department of Computer Science and Engineering

Jadavpur University, Kolkata-700032

India

2019

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Certificate of Recommendation

This is to certify that the dissertation entitled "Leaf Classification using Convolutional Neural Network" has been carried out by Sayantan Maity (University Registration No.: 140763 of 2017-2018, Examination Roll No.: M4CSE19008) under my guidance and supervision and be accepted in partial fulfillment of the requirement for the Degree of Master of Computer Science and Engineering. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree in any other University or Institute.

....

Dr. Susmita Ghosh (Thesis Supervisor)

Department of Computer Science and Engineering

Jadavpur University, Kolkata-32

Countersigned

....

Prof. Mahantapas Kundu

Head, Department of Computer Science and Engineering Jadavpur University,

Kolkata-32.

....

Prof. Chiranjib Bhattacharjee

Dean, Faculty of Engineering and Technology Jadavpur University,

Kolkata-32.

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

<u>Certificate of Approval</u>

This is to certify that the thesis entitled "Leaf Classification using Convolutional Neural Network" is a bona-fide record of work carried out by Sayantan Maity in partial fulfillment of the requirements for the award of the degree of Master of Computer Science and Engineering in the Department of Computer Science & Engineering, Jadavpur University during the period of June 2018 to May 2019. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

…………………………………………………………………………………..
Signature of Examiner 1 Date:

…………………………………………………………………………………..
Signature of Examiner 2 Date:

*Only in case the thesis is approved

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis entitled "Leaf Classification using Convolutional Neural Network" contains literature survey and original research work by the undersigned candidate, as part of his Degree of Master of Computer Science and Engineering.

All information have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: Sayantan Maity

Registration No: 140763 of 2017-2018

Exam Roll No.: M4CSE19008

Thesis Title: Leaf Classification using Convolutional Neural Network

…..……………………………………….

Signature with Date

Acknowledgement

I would like to start by thanking the holy trinity for helping me deploy all the right resources and for shaping me into a better human being.

I would like to express my deepest gratitude to my advisor, Prof**. Susmita Ghosh**, Department of Computer Science and Engineering, Jadavpur University for her admirable guidance, care, patience and for providing me with an excellent atmosphere for doing research. Our numerous scientific discussions and her many constructive comments have greatly improved this work.

I am immensely grateful to my lab mates Mr. Utshab Saha, and Miss Payel Pramanik who were always present to motivate, guide me during the result analysis process. Without all of them it would surely be a very lonely lab. I sincerely thanks my seniors Miss Tithi Bhakta and Mr. Ranajoy Banerjee for their support in completion of the thesis documentation.

Most importantly none of this would have been possible without the love and support of my family. I extend my thanks to my parents, especially to my father whose forbearance and whole hearted support helped this endeavor succeed. This thesis would not have been completed without the inspiration and support of a number of wonderful individuals — my thanks and appreciation to all of them for being part of this journey and making this thesis possible.

……………………………………………..

Sayantan Maity

Exam Roll No.: M4CSE19008

Registration No: 140763 of 2017-2018

Department of Computer Science & Engineering

Jadavpur University

Abstract

In this work a study has been made on leaf classification. To do this work, we have proposed a new convolution neural network (CNN) having different architecture. Five different models are presented having five different architectures. The entire details of different parameters i.e., the number of layers, various activation functions, different optimization functions for all the models has been described and the comparative analysis i.e., the effectiveness of these parameters has been made. Addition of two new layers is done in one model. It is seen that, addition of a batch normalization layer before a leaky-relu activation layer and addition of a dropout layer after this activation layer in the CNN model architecture prevents the over fitting problem of the model, where the four models cannot.

Date :

Place :

Sayantan Maity

Exam Roll No.: M4CSE19008

Registration No: 140763 of 2017-2018

Department of Computer Science & Engineering.

Jadavpur University.

Chapter 1

Introduction

1.1 Classification

Classification is a process in which an object is categorized, recognized, or differentiated. In context of machine learning, the task, classification is to find a hypothesis H, where the hypothesis H is used for discrimination of different objects. This hypothesis is called as classifier. There are different types of methodologies for building automated classifier, where the classifiers learn from the objects itself. Support Vector Machine [2], K-Nearest Neighbors (KNN) [3], Artificial Neural networks [4] are some popular classifiers in the machine learning domain.

1.2 Image Classification

Image classification refers to a process in computer vision, where classification of images are done depending on its visual content (i.e., some attributes like pixel values, edges, spectral information, and spatial information of an image). For example, an image classification algorithm can be designed in a way so that it can differentiate between the image of a tree and an image of human face. Though several techniques are there for classification, robust image classification is still a challenge in computer vision applications.

1.3 Leaf Classification

Leaf is a very important feature or characteristics of a tree for classification. Trees can be identified by its leaf. Leafs differs from each other by its shape, size, color, texture etc. Besides, leafs are characterized by some features. In computer vision problem, leaf classification is a challenging task. To computer an Image is a matrix of pixel values. Pixel values are very low level features of an Image. As computers cannot comprehend images, they are required to be converted into features by individually analyzing image shapes, colors, textures and moments. Images that looks similar may deviate in terms of geometric and photometric variations. So it is required to extract meaningful high level features from a given leaf image. Classification accuracy is heavily dependent on the features that represents the leaf. It can be said that, the good or important features have been extracted, if it has low variance in leafs of same class and high variance in leafs of different class and if it is able to extract the features with less error from a given leaf image.

1.4 Application of Leaf Classification

Plants are fundamentally important to life. Key research areas in plant science include plant species identification, monitoring plant health and tracing leaf growth, and the semantic interpretation of leaf information. Botanists easily identify plant species by discriminating between the shape of the leaf, tip, base, leaf margin and leaf vein, as well as the texture of the leaf and the arrangement of leaflets of compound leaves. Because of the increasing demand for experts and calls for biodiversity, there is a need for intelligent systems that recognize and characterize leaves so as to scrutinize a particular species, the diseases that affect them, the pattern of leaf growth, and so on.

1.5 Need of Automated Feature Extraction

The common approach of image classification is utilizing one of the conventional classification methods such as support vector machines and decision trees on extracted features, which are transformed from raw images. The dimensionality of raw image data is simply too high for most classification methods. Furthermore the spatial relationships between neighboring pixels cannot be observed by these methods since they often consider data points independent from each other.

Therefore the feature extraction phase in conventional approaches is critical. The subsequent classification is directly affected by the extracted features. Poor features would not lead to accurate classification.

The challenges in feature extraction reside in several aspects. The goodness of a certain feature is often problem dependent. For a specific application, designing a totally new feature may be required. There are no universal features which can excel in all applications. Also there are numerous existing image features such as histogram features, edginess features, texture features and features in the frequency domains. Deciding a suitable set of features is usually the focus of image classification applications. Such task heavily depend on domain knowledge as the understanding of the task itself and the extensive experience on existing features in the literature are crucial[5].

1.6 Scope of the Thesis

In this work a study has been made on leaf classification. To do this work, we have proposed convolution neural network (CNN) having different models or architectures. So in this thesis the entire details of different parameters i.e., the number of layers, various activation functions, different optimization functions for all the models has been described and the comparative analysis i.e., effectiveness of these parameters has been made. Addition of two new layers is done in one model. It is seen that, addition of a batch normalization layer before a Leaky-ReLU activation layer and addition of a dropout layer after this activation layer in the CNN model architecture prevents the over fitting problem of the model. Also significant accuracy improvement in validation dataset is seen.

1.7 Organization of the Thesis

In Chapter 2 we describe brief introduction of Convolution neural network and its different component. In Chapter 3 we discuss different features of leaf data and also technique to extract those feature and classification technique exist in literature. Dataset descriptions, results and their analysis of the work are detailed in chapter 4 and conclusion is described in Chapter 5.

Chapter 2

Convolution Neural Network

2.1 Introduction

In Convolution Neural Network convolution operation is used to produce a feature vector to feed the fully connected layers (classification layer). The network consists of a stack of consecutive convolution layers followed by dense layers. Convolutional Neural Network (CNN) was first introduced in computer vision for image recognition by LeCun et al. in 1989[6]. Since then, it has been widely used in image recognition and classification tasks. The recent impressive success of Krizhevsky et al. in ILSVRC 2012 competition [7] demonstrates the significant advancement of modern deep CNN on image classification problem [8]. Inspired by his work, many recent research works have been concentrating on understanding CNN and extending its applications to conventional computer vision tasks.

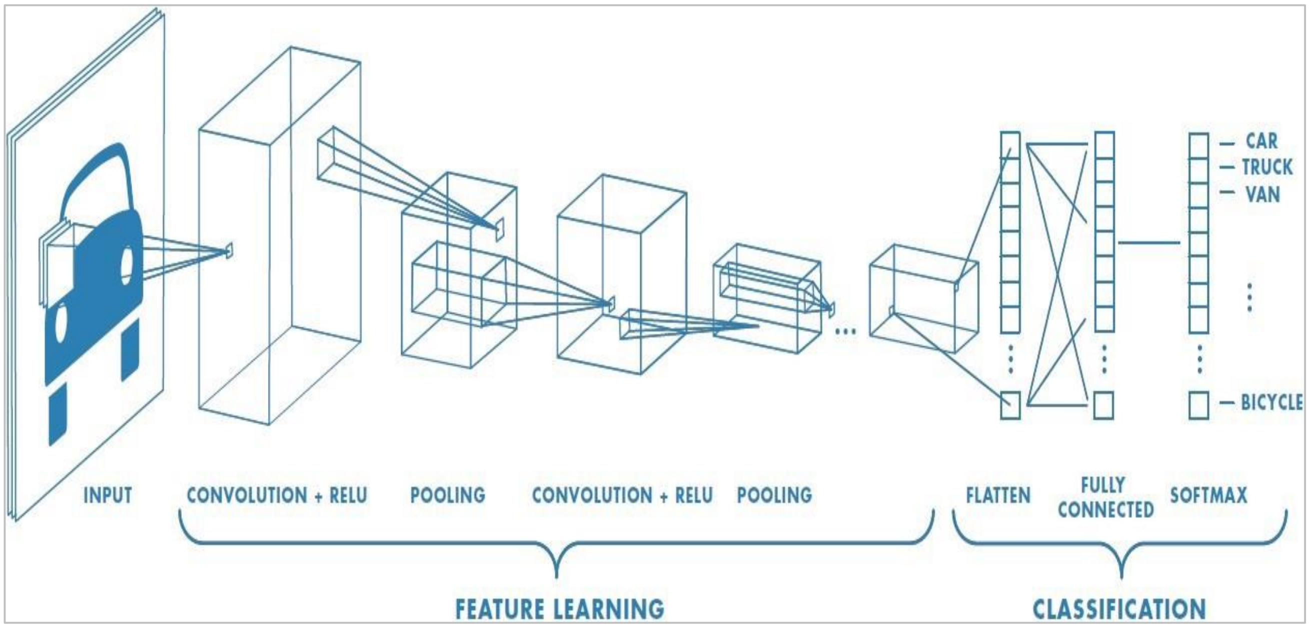


Fig 2.1: Visual diagram of a CNN

2.2 Convolution Operation:

Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel. It should be noted that the matrix operation being performed, convolution, is not traditional matrix multiplication, despite being similarly denoted by *. It is a matrix dot product.

For example, if we have two three-by-three matrices, where the first one is a kernel, and the second is an image piece, convolution is the process of flipping both the rows and columns of the kernel and then multiplying locally similar entries and summing up. The element at coordinates [2, 2] (i.e., the central element) of the resultant image would be a weighted combination of all the entries of the image matrix.

The other entries would be similarly weighted, where we position the center of the kernel on each of the boundary points of the image, and compute a weighted sum. The value of a pixel in the output image is calculated by multiplying each kernel value by the corresponding input image pixel value.

The general expression of a convolution is:

$$g(x,y) = \omega * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t),$$

Where *g(x, y)* is the filtered image, *f(x, y)* is the original image, *w* is the filter kernel. Every element of the filter kernel is considered by (-a<=s<=a) and (-b<=t<=b.) Depending on the element values, a kernel can cause a wide range of effects, it is shown in the figure below.

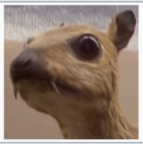
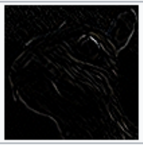
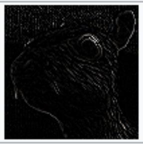
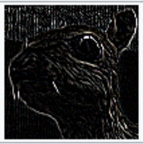
Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Fig 2.2: Effect of Feature Detection Kernel

2.3. Convolution Layer

At convolution layer, a linear operation is done that involves the multiplication of a set of weights with the input, as we do in traditional neural network. A technique was designed for two-dimensional input, multiplication is performed between an array of input data and a two-dimensional array of weights, called a filter or a kernel.

The filter is made smaller than the input data and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product. A dot product is the element-wise multiplication between the filter-sized patch of the input and filter, which is then summed, always resulting in a single value. Since it results in a single value, the operation is often referred to as the "*scalar product*".

Using a filter smaller than the input is intentional as it allows the same filter (set of weights) to be multiplied by the input array multiple times at different points on the input. Specifically, the filter is applied systematically to each overlapping part or filter-sized patch of the input data, left to right, top to bottom.

If the filter is designed to detect a specific type of feature in the input, the application of that filter across the entire input image allows the filter an opportunity to discover that feature anywhere in the image. This capability is commonly referred to as translation invariance that is the general interest is in whether the feature is present or not rather than where it was present.

As the filter is applied multiple times to the input array, the result is a two-dimensional array of output values that represent a filtering of the input. As such, the two-dimensional output array from this operation is called a "*feature map*"[6]. Once a feature map is created, we can pass each value in the feature map through a nonlinear activation function, such as a ReLU, much like we do for the outputs of a fully connected layer.

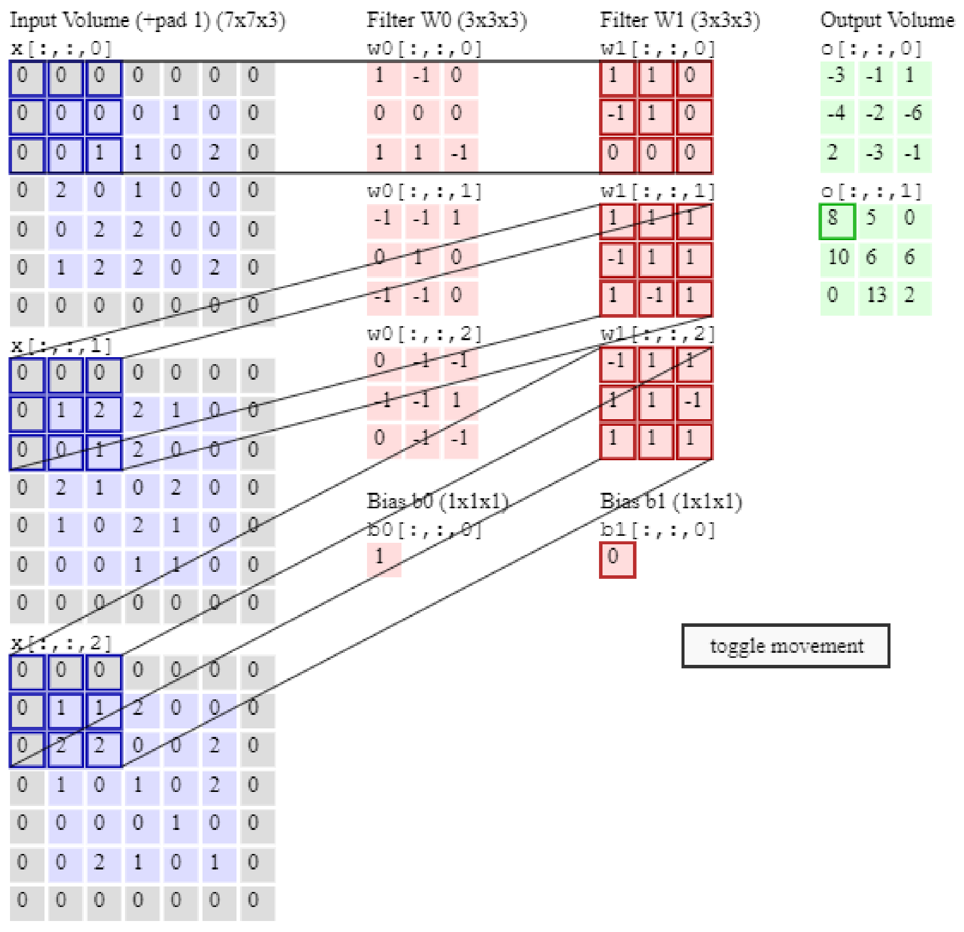


Fig 2.3: Convolution Operation

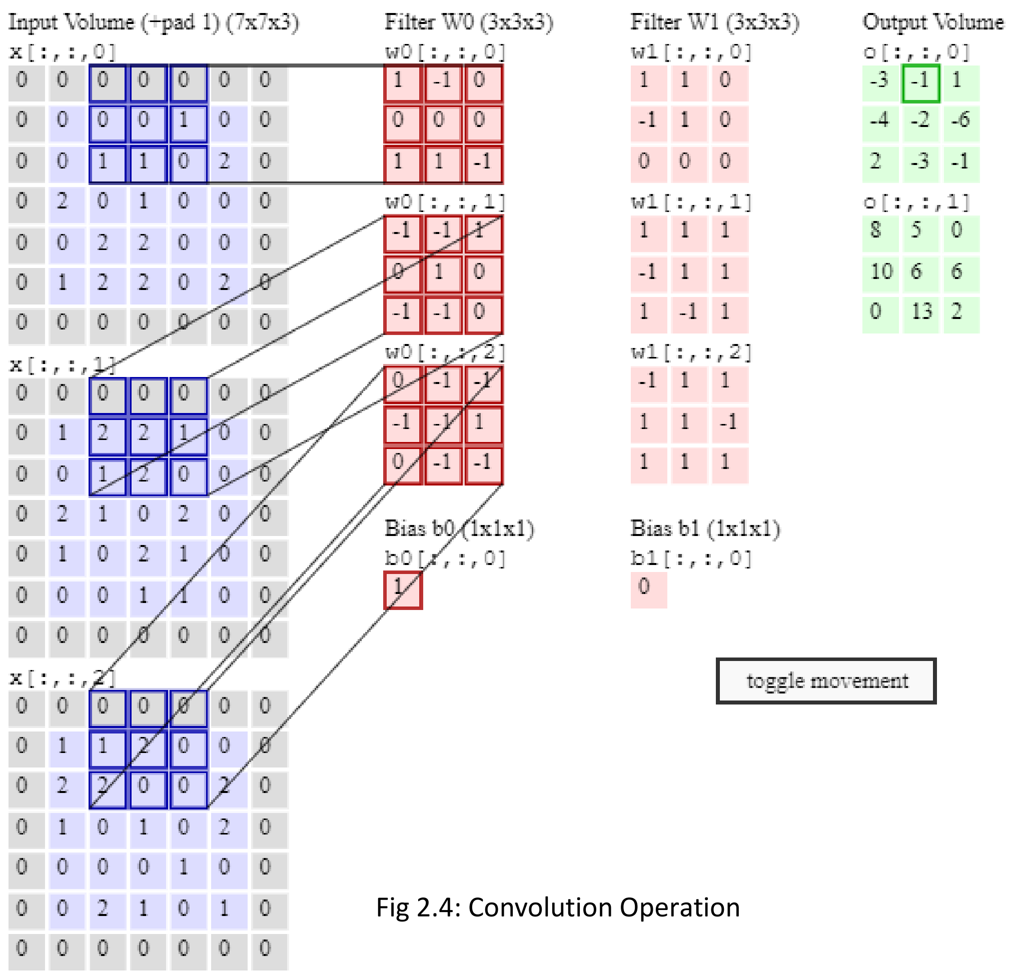


Fig 2.4: Convolution Operation

Fig. 2.3 & Fig. 2.4 are examples of convolution operation on a 7x7 image by two 3x3 kernels, padding size = 1 and stride = 2 with bias. Padding is an additional layer that we can add to the border of an image, for example, suppose we add one more layer to a 4*4 image and convert it in to a 5*5 image for the sake of accuracy, then this layer of numerical zeroes is called zero padding.

2.3.1 Kernel Size

The size of kernel is a hyper-parameter of CNN architecture [ ]. The area of kernel is receptive field of activation map. We usually define kernel only by its spatial size (*k x k),* where k is length and width of the kernel, for example (2 x 2), (3 x 3), (7 x 7). Kernel size always smaller than input size. Conventionally we consider kernel size as an odd number. Actual size of kernel is kernel spatial dimension with number channels of input to the convolution layer. If we consider number of channel of input is input_channel the kernel shape is (k *x k x input_channel*). For an example, if input is RGB image to first layer CNN then kernel size will be

k x k x 3, where *k* is length or width of the kernel.

2.3.2 Number of Kernels

Number of kernel is also a hyper-parameter of convolution a layer. Different kernel is used to extract different image feature. Depth or number of channel in an output feature map is equal to number of kernel.

2.3.3 Bias

Use of bias with a kernel is also a choice for a network. A bias is single variable which is added to convolve value of kernel and pixel values of the current receptive field. Reason behind using bias is to shift activation position. If a non-linear activation function used to in convolution layer then it is good to use bias.

2.3.4 Activation Functions

To make non-linear feature map, convolved value is pass through a nonlinear activation function.

- Sigmoid
  The sigmoid function (Fig. 2.5) is defined as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

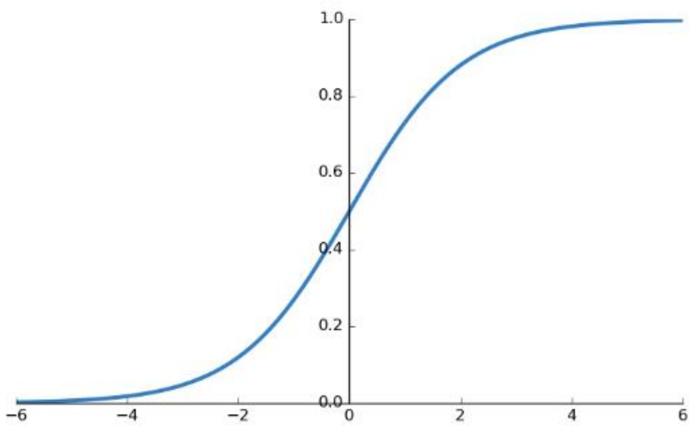


Fig 2.5: Sigmoid Function

Local gradient of sigmoid

$$f'(x) = f(x)(1 - f(x))$$

This a non-linear function that suffers from saturation [ ].

Saturation of activation

An activation that has an almost zero gradient at certain regions. This is an undesirable property since it results in slow learning or ensure vanishing gradient when use in deep network.

- **tanh**

  The tanh function (Fig 2.6) is defined as:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{-x} + e^{-x}}$$

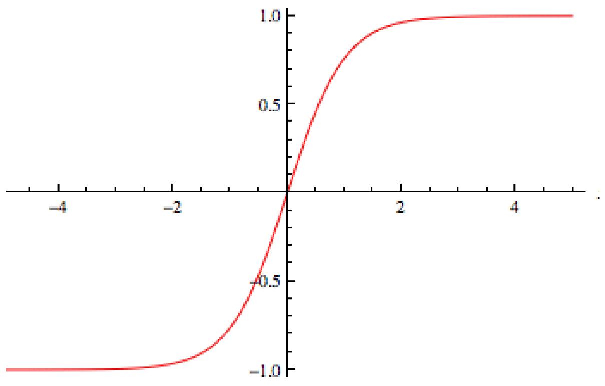


Fig 2.6: tanh Function

It has a similar problem like sigmoid activation function.

The only difference is that it has zero center output, and range is from (-1, 1).

Local gradient of tanh is:

$$f'(x) = 1 - f(x)^2$$

- **ReLU**

Rectified Linear Unit or ReLU (Fig. 2.7) is defined as:

$$f(x) = \max(0, x)$$

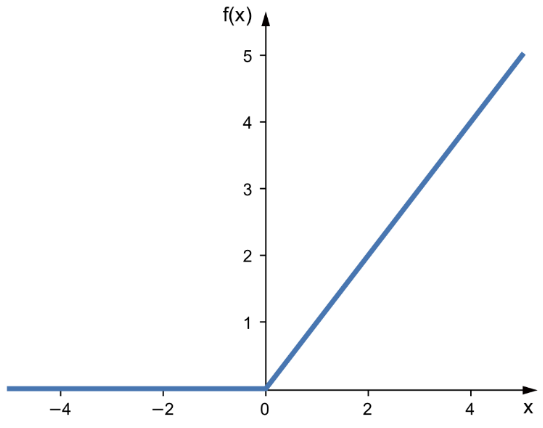


Fig 2.7: ReLU Function

- **Leaky-ReLU**

Leaky-ReLU activation function (Fig. 2.8) is defined as follow

$$f(x) = \max(\gamma * x, x) \ \ where\ 0.0 \leq \gamma \leq 1.0$$

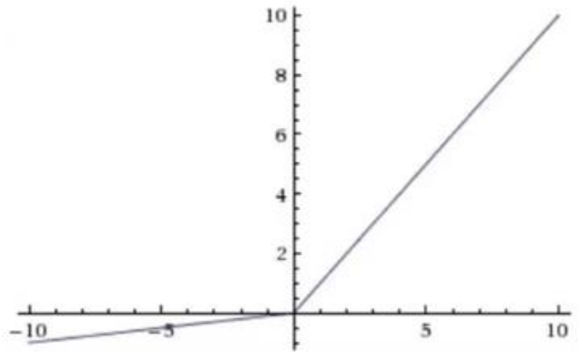


Fig 2.8: Leaky-ReLU Function

The Local gradient is:

$$f'(x) = \max(\gamma\ , 1)$$

In ReLU, gradient flow for negative input is zero. So ReLU may have gradient vanishing problem (which is zero) but Leaky-ReLU is free from this problem. Moreover if a kernel gives negative output and we use ReLU then the output of feature map is always same and zero for any input and the local gradient is also zero; so kernel will never be able to update through back propagation. Kernel will be a useless or dead kernel.

2.3.4 Stride

Stride of kernel is also a hyper parameter. Stride is a number that a kernel skip or slide through a direction to produce next value of a feature map.

Size of a feature map or output depends on stride value. Usually we take stride value of a kernel is 1. It is always less or equal to kernel size.

2.3.5 Number of Learnable Parameters & Output Size of a Convolution Layer

Let,

- Kernel size = k x k
- Number of kernels = d
- Stride = s
- Input shape = (n, m, c) where n = height, m = length, c = channel
- Then, output shape $=(\frac{n-k+1}{s}, \frac{m-k+1}{s}, d)$
- Number of learnable parameters $= (k * k * c + 1) * d$

2.4 Sub-sampling or Pooling Layer

The pooling layer is used to reduce the spatial dimensions. On a convolution neural network, uses of pooling layer provides the following advantages:

- By having less spatial information, computation performance increased
- Less spatial information also means less parameters, so less chance to over-fit
- Network gets some translation invariance

In the Figure below (Fig. 2.9) it is shown, that the most common type of pooling, the max-pooling layer, which slides a window, like a normal convolution, and get the biggest value on the window as the output.

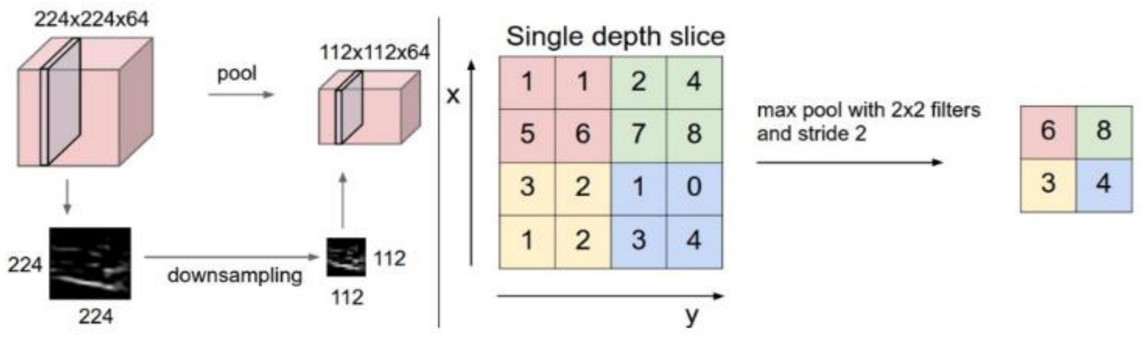


Fig 2.9: Max-pooling Operation

In this pooling layer window or patch size is also a hyper parameter. Usually we always take patch size 2 x 2 with stride value 2. Every patch on the given input feature map will reduce to one pixel value.

The most important parameters in pooling layer are:

- Input shape = (n x m x c); where n = height, m = length, input_channel = c
- Stride: s; Scalar that controls the amount of pixels that the window slides.
- Kernel size = k
- Output shape : w1 x h1 x c

Where, $w1 = \frac{(n-k)}{s} + 1$ and $h1 = \frac{(m-k)}{s} + 1$

In the pooling layer there is no learnable parameters. So back propagation becomes simpler.

2.4.1 Max-pooling

It passes the max*imum* value among the pixels of the given patch area or current window.

2.4.2 Avg-pooling

It passes the average value of pixels of for a given patch area or window.

2.4.3 Min-pooling

It passes the min value among the pixels of the given patch area.

2.5 Dropout Layer

Dropout is a technique where randomly selected neurons are ignored during training (Fig. 2.10). They are “dropped-out” randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass. As a neural network learns, neuron weights settle into their context within the network. Weights of neurons are tuned for specific features providing some specialization. It can be imagined that if neurons are randomly dropped out of the network during training, that other neurons will have to step in and handle the representation required to make predictions for the missing neurons. The effect is that the network becomes less sensitive to the specific weights of neurons. This in turn results in a network that is capable of better generalization and is less likely to overfit the training data. During the prediction phase the dropout is deactivated.

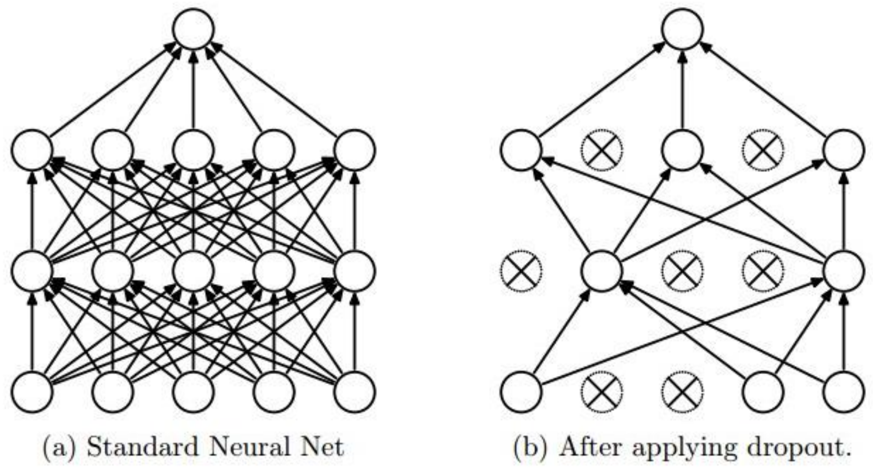


Fig 2.10 (a) Standard and Dropout Neural Net

Normally some deep learning models use Dropout on the fully connected layers, but is also possible to use dropout after the max-pooling layers, creating some kind of image noise augmentation. The probability of a neuron to be dropped is considered as the only hyper parameter in drop out layer.

2.6 Batch Normalization Layer

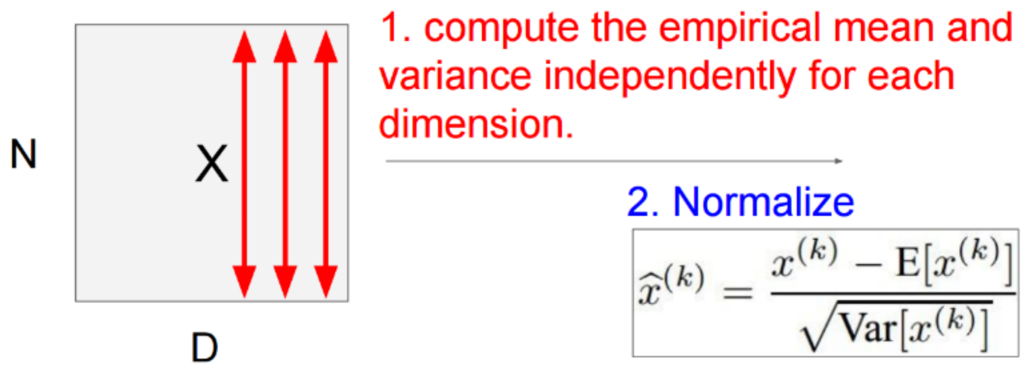
Normalization is done on the activation of every fully connected layer or Convolution layer during training. Batch-normalization is viewed as an adaptive (or learnable) pre-processing block with trainable parameters which are tuned through back propagation. The activation function of each fully connected layer is normalized, generally batch normalization is done using mean and standard deviation values.

Some advantages of using Batch-Norm:

- Improves gradient flow, used on very deep models.
- Allows higher learning rates
- Reduces dependency on initialization
- Gives some kind of regularization (so Dropout loses some importance but we keep using it).

This forces activations (Convolution layer, Fully-connected layer outputs) to become unit valued standard deviation and zero mean.

To each learning batch of data we apply the following normalization.


N
X
D
1. compute the empirical mean and variance independently for each dimension.
2. Normalize
$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$

The output of the batch normalization layer, has three parameters. Those parameters will be learned to best represent our activations. Those parameters allow two learnable (scale and shift) factor.

Now summarizing the operations:

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

2.7 Gradient Descent

Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks. At the same time, every state-of-the-art Deep Learning library contains implementations of various algorithms to optimize gradient descent .These algorithms, however, are often used as black-box optimizers [ ].

- Gradient descent variants

There are three variants of gradient descent, which differ in how much data we use to compute the gradient of the objective function. Depending on the amount of data, we make a trade-off between the accuracy of the parameter update and the time it takes to perform an update.

2.7.1 Batch Gradient Descent

Vanilla gradient descent or batch gradient descent [ ] computes the gradient of the cost function $J(\theta)$w.r.t. to the parameters θ for the entire training dataset:

$$\theta = \theta - \gamma * \nabla\theta\, J(\theta)$$

Where, θ is learnable parameters (Weights of neurons) of network and γ is learning rate.

As we need to calculate the gradients for the whole dataset to perform just one update, batch gradient descent can be very slow and is intractable for datasets that do not fit in memory. Batch gradient descent also does not allow us to update our model online, i.e. with new examples readily.

In code, batch gradient descent looks like:

```
for i in range ( num_epochs ):
    params_grad = evaluate_gradient ( loss_function , data , params )
    params = params - learning_rate * params_grad
```

For a pre-defined number of epochs, at first the gradient vector params_grad of the loss function for the whole dataset w.r.t. our parameter vector params is computed. An important point which is to be noted here is that state-of-the-art deep learning libraries provide automatic differentiation that efficiently computes the gradient w.r.t. some parameters. If the gradients are derived separately by a user, then in that case gradient checking is a good idea. The parameters are then updated in the direction of the gradients with the learning rate determining how big of an update is to be performed. Batch gradient descent is guaranteed to converge to the global minimum for convex error surfaces and to a local minimum for non-convex surfaces. Different types of optimization approach like Graph Computation, TensorFlow ML framework eases our work of doing gradient computation.

2.7.2 Stochastic Gradient Descent:

Stochastic gradient descent (SGD), in contrast, performs a parameter update for each training example $x(i)$ and label $y(i)$:

$$\theta = \theta - \gamma * \nabla\theta \, J\big(\theta; x(i); y(i)\big) \qquad \ldots (2)$$

As discussed in [ ].

Batch gradient descent performs redundant computations for large datasets, as it re-computes gradients for similar examples before each parameter update [ ]. SGD does away with this redundancy by performing one update at a time. It is therefore usually much faster and can also be used to learn online. SGD performs frequent updates with a high variance that cause the objective function to fluctuate heavily. While batch gradient descent converges to the minimum of the basin the parameters are placed in, SGD's fluctuation, on the one hand, enables it to jump to new and potentially better local minima. On the other hand, this ultimately complicates convergence to the exact minimum, as SGD will keep overshooting. However, it has been shown that when we slowly decrease the learning rate, SGD shows the same convergence behavior as batch gradient descent [ ], almost certainly converging to a local or the global minimum for non-convex and convex optimization respectively. Its code fragment simply adds a loop over the training examples and evaluates the gradient w.r.t. each example. Note that we shuffle the training data at every epoch [ ].

2.7.3 Mini-batch Gradient Descent

Mini-batch gradient descent takes the best of both worlds and performs an update for every mini-batch of *n* training examples:

$$\theta = \theta - \gamma * \nabla J\big(\theta; x(i : i+n); y(i : i+n)\big) \qquad \ldots (3)$$

This way, it a) reduces the variance of the parameter updates, which can lead to more stable convergence; and

b) Can make use of highly optimized matrix optimizations common to state-of-the-art deep learning libraries that make computing the gradient w.r.t. a mini-batch very efficient. Common mini-batch sizes range between 50 and 256, but can vary for different applications. Mini-batch gradient descent is typically the algorithm of choice when training a neural network and the term SGD usually is employed also when mini-batches are used.

2.7.4 Gradient Descent Optimization Algorithms

In the following, we will outline some algorithms that are widely used by the Deep Learning community to deal with the aforementioned challenges. We will not discuss algorithms that are infeasible to compute in practice for high-dimensional data sets, e.g. second-order methods such as Newton's method.

Momentum SGD

Momentum SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another, which are common around local optima. In these scenarios, SGD oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local optimum.

Momentum is a method that helps accelerate SGD in the relevant direction and dampens oscillations. It does this by adding a fraction α of the update vector of the past time step to the current update vector.

$$v * t = \alpha * v * t - 1 + \gamma * \nabla\theta * J(\theta) \quad \dots (4)$$

$$\theta = \theta - v * t \quad \dots (5)$$

The momentum term α is usually set to 0.9 or a similar value. Essentially, when using momentum, we push a ball down a hill. The ball accumulates momentum as it rolls downhill, becoming faster and faster on the way (until it reaches its terminal velocity, if there is air resistance, i.e. $\alpha < 1$). The same thing happens to our parameter updates: The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, we gain faster convergence and reduced oscillation.

Chapter 3

Literature Survey: Leaf Classification

3.1 An Overview of Leaf Classification Systems

The researchers Du et al. [6] have analyzed morphological features and invariant moment features of various shapes of different plant databases and applied the move median centers (MMC) hyper sphere classifier to classify leaf species. They used a leaf database containing only a single leaf image against a blurred background, and collected a total of 20 species of different images with a total of 400 scanned leaf images. Macleod et al. [7] investigated several computer-assisted systems for the species identification of living and nonliving things based on the DNA bar-coding scheme. They studied systems in oceanographic-based research and paleontology, and tested his work in the Digital Automated Identification System (DAISY), classifying only 30 species. They worked on din flagellate categorization using the Artificial Neural Network (DiCANN) system to identify phytoplankton species with 72% accuracy. Pattern Analysis, Statistical Modeling and Computational Learning (PASCAL) were used to classify common objects. A plant species identification system for the broad leaves found in Norway was proposed by Babatunde et al. [7] which were based on the morphological features of the leaves and they also discussed different features of leaves and feature extraction techniques. In [8], various leaf structures and flower feature extraction techniques and problems in an agricultural environment were reviewed. Detailed information of the important survey paper. We have identified papers that have discussed feature extraction techniques, as well as those that included classification techniques, those based on particular leaf species, those that included a combination of shape and venation, those that included a combination of texture and texton, and those that worked to resolve the problem of big data. We list here the paradigms used for our study, and Table 2 shows the number of papers included for this detailed analysis so as to handle different problems in agricultural research.

P1: Analysis based on different leaf shapes

P2: Analysis based on venation

P3: Analysis based on leaf tip, base, and margin

P4: Analysis based on texture/texton

P5: Analysis based on moment invariant descriptors

P6: Analysis based on different classification techniques to resolve problems with inter and intra-class classification, imbalanced data, and managing big data.

P7: Analysis based on different leaf databases

3.2 Block Diagram of Leaf Classification System

The general block diagram of leaf species identification system is shown in *Fig. 3.1*. In this system a user gets the leaf image to be identified. Then the system performs image preprocessing such as conversion of a color image to grayscale image, image smoothing by removing noise, segment the images etc. Next, the system extracts the general features of leaf such as shape, color, texture and some of the leaf specific features such as leaf tip, base, apex and margin and venation information. These features are compared with the features of the leaves stored in a database to identify the species of the leaf based on Intra and inter classes' similarity.

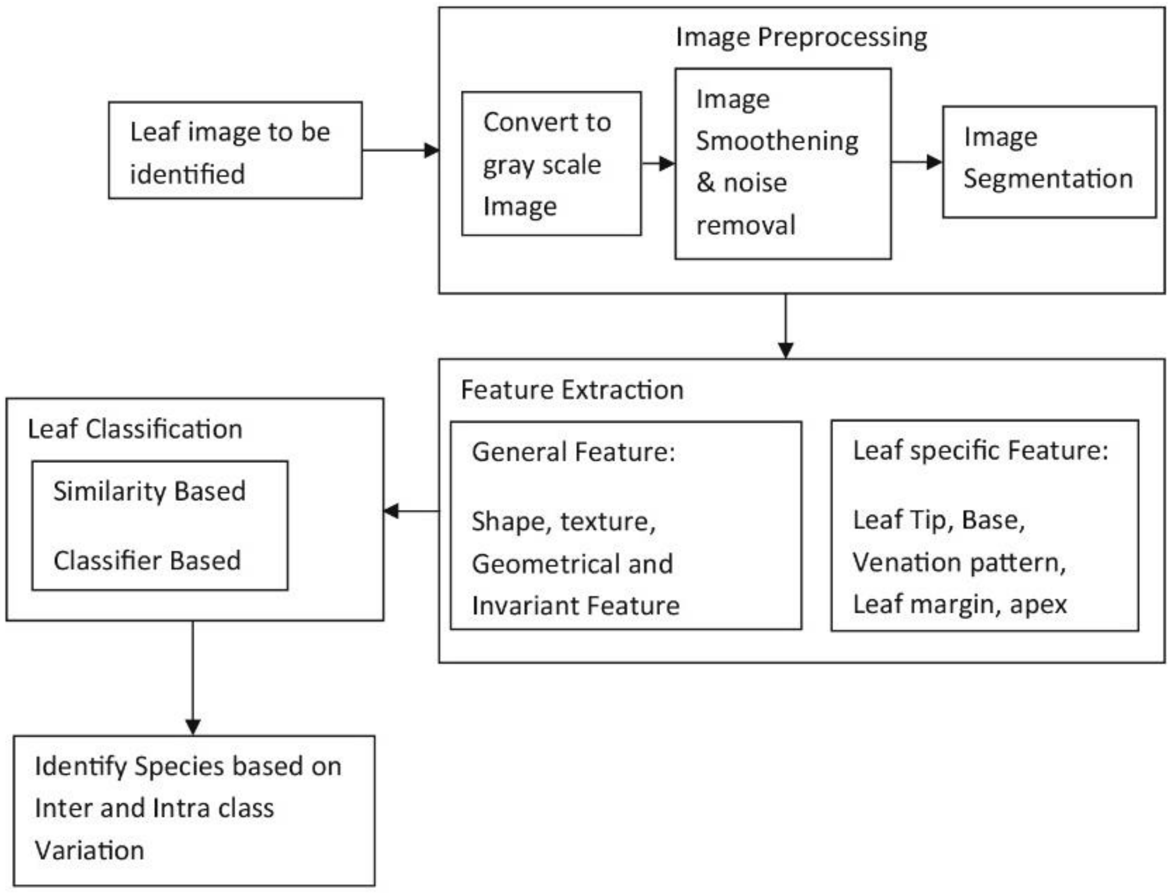

Leaf image to be identified
Image Preprocessing
Convert to gray scale Image
Image Smoothening & noise removal
Image Segmentation
Feature Extraction
General Feature:
Shape, texture, Geometrical and Invariant Feature
Leaf specific Feature:
Leaf Tip, Base, Venation pattern, Leaf margin, apex
Leaf Classification
Similarity Based
Classifier Based
Identify Species based on Inter and Intra class Variation

Fig. 3.1. Block Diagram of Leaf species identification system

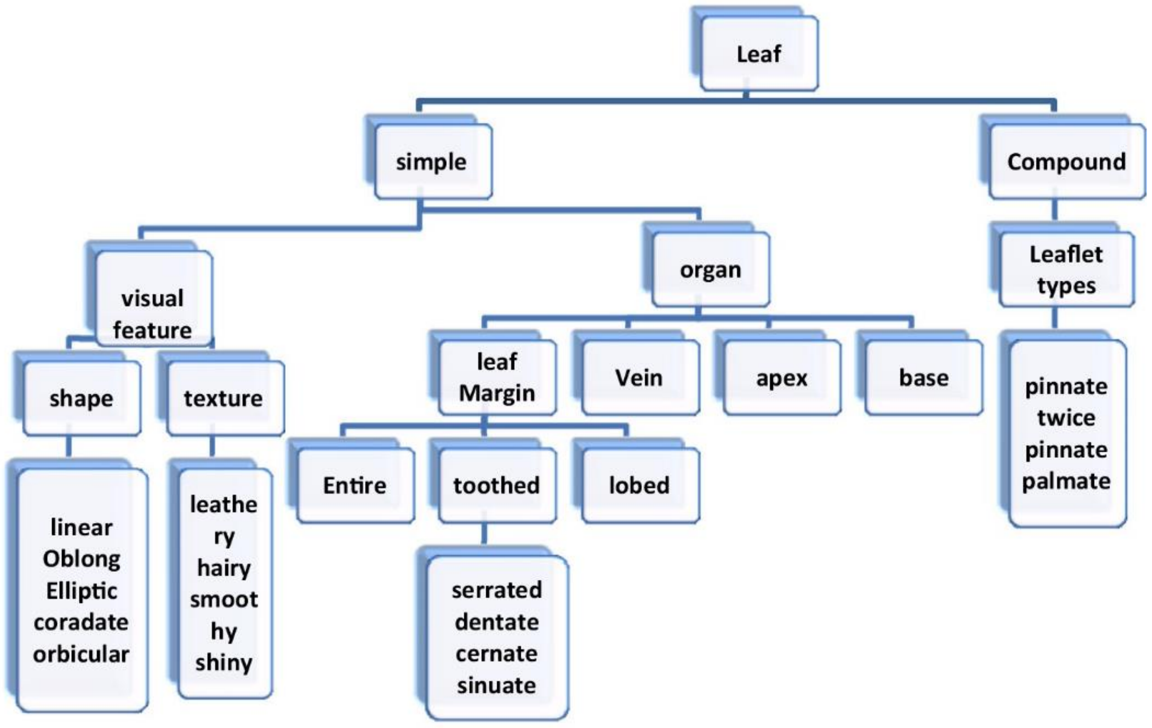

Leaf
simple
Compound
visual feature
organ
Leaflet types
shape
texture
leaf Margin
Vein
apex
base
pinnate twice pinnate palmate
Entire
toothed
lobed
linear Oblong Elliptic coradate orbicular
leathery hairy smooth y shiny
serrated dentate cernate sinuate

Fig 3.2 Types of leaf features

3.3 State-of-the-Art Techniques in Feature Extraction

A feature is a piece of information relevant to a specific leaf image, and is divided into two types: local and global. Local features are extracted from leaf patches and global features from leaf shape, texture and color. All leaves are identical in terms of color, which can vary with climatic changes. Color, shape and texture are appropriate features for the classification of leaf species. There are two types of leaves: simple and compound leaves, according to leaf manual [15] their general structures are as shown in Fig. 3.2 Cope et al. [9] discussed the morphological structure of simple leaves, which are identified through key features, such as color, shape, margin, venation, and arrangement. Compound leaves, however, are identified by the number of leaflets in a stalk, with the extraction of feature from single leaflet. There is, therefore, a need for appropriate features for the identification of leaf species. Sharma and Gupta [11] presented an overview of some of the common methods used for leaf feature extraction and classification.

3.4 Feature Extraction Techniques

Feature extraction is an important technique used in image classification, pattern recognition and object recognition. In order to have effective classification of plant species researchers should decide to extract efficient features. Researchers classify plants using roots, fruits, seeds and flowers [13–14]. Leaf color [15] cannot be considered a viable feature for classification because it may vary with climatic and camera calibrations. Given that most leaves are green, they are to be classified through shape, texture and invariant feature descriptors that are invariant to translations, rotations and scaling transformations

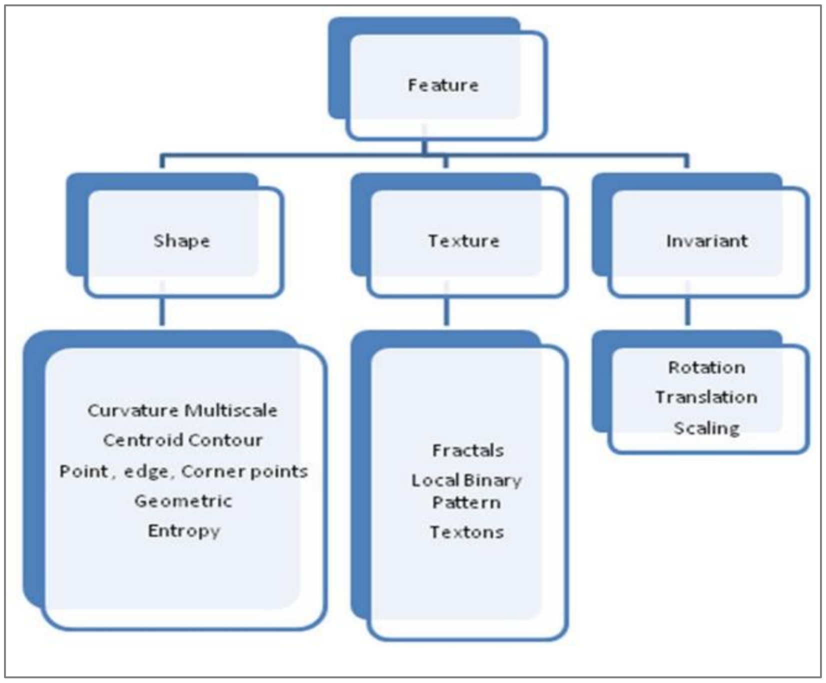


Fig 3.3 Feature extraction techniques

of images. Since color is not considered, grayscale images of leaves are used for identification. Figure 3.3 shows different feature extraction techniques.

3.4.1 Curvature Descriptors

Curvature Scale Space (CSS) is a technique used to measure the contours of shapes, extracts the concavity and convexity of curvature. It is invariant to translation and rotation in a viewpoint direction but not in scale, because it varies with the Gaussian kernel (a) and cannot easily fix the value of the Gaussian kernel. It leads to misclassification of serrated and lobe-shaped leaves. Curvature is a vital property of leaves and curvatures are computed using differential techniques. However, it produces more noise, is sensitive to rotation, and generates different feature vectors with different scales. It is impossible to sustain all the curvature features combined together in one feature vector. Aligning them all in one particular point is a difficult task, because the features differ for each scale.

According to [16] CSS used to identify the starting and ending points of the venation feature points of leaves by estimating the maximum angles of the leaves. The densities of feature points are estimated using the Parzen window method for non-parametric density estimation and it can be applied to any data distribution. We cannot, however, get to choose the correct window size. According to [17], since veins are represented as strings used for semantic interpretation, there is no need to find the starting and ending points. But these methods cannot be used for imperfect and overlapped leaves. Grinblat et al. [18] used an unconstrained hit or miss transform technique to extract particular patterns in foreground and background pixels. When applied to leaf images, central vein patches are extracted from leaves and various geometric features are calculated for the veins. The SIFT descriptor [19, 20] were used to extract key features from an image. It produces good results on the circular orientation of an image, and is well suited to illumination and various viewing conditions. It extracts histogram features from local patches. The authors extracted corner points using Mean Projection transform (MPT) instead of CSS, it produces indistinguishable variations as well as aliasing. To eliminate such problems, the Mean Projection Transform extracts corners that have high curvature. The Flavia dataset produces accuracy of 87.5%. The researchers Chen et al. [21] proposed a velocity representation technique to represent curvature points. This algorithm computes only 9 points on a leaf contour. It reduces the running time of the algorithm, because the CSS computes 200 intersection points on the curvature and increases the running time. Square root velocity representation [22] was used for shape-based leaf classification to solve the intra-class and inter-class variability of leaf images. It automatically detects similarities by computing the geo desic distance of statistical shape features and 2D planar curves by computing the elastic deformations of the Riemannian structure. It is applied to the Flavia leaf dataset.

3.4.2 Multi-scale Descriptors

The multi-scale descriptors furnish much more information about leaf contours. Derived from the scale space and image pyramid structure, it extracts image features at various levels by capturing local and global features from low- to high-resolution scales. It provides the maximum discriminating power and is robust to noise depending on the boundaries of leaves and not the regions of an image. As a result, it works well on feature space rather than image space. Multi-scale Triangular Area Representation (MTAR) is used in [23] which is affine invariant, robust to noise and provides the features of images concavity and convexity. He also developed triangle side length and triangle-oriented angle descriptors for leaf images. The researchers Wang et al. [24] introduced Multi-scale Arc Height Descriptor (MARH) which is invariant to translation. It enumerates a local normalization technique for each scale to employ rotation and scaling, because the local normalization rendered for each scale is based on the maximum value of arch height descriptors. It leads to shape dissimilarity at each different scale, so is invariant to translation and scaling. It measures the arch height of palmate-shaped and lobe-shaped leaves but is unsuitable for overlapped leaves. In this method, the local normalization scheme is applied for scaling and rotation. It takes longer execution time, compared to other invariant descriptors. A new method called the Multi-scale Bending Energy (MBER) was proposed by Souza et al. [25] which require energy to perform at the lowest energy rate on a curvature signal based on its sensitivity to the local features of the shape contour. It provides low noise immunity and spatial locations of certain prominent points. Given these limitations, its use in shape description is rather limited. Researchers of papers [26] used curvelet transform, which is a multi-scale object representation technique applicable only to objects with small length scales. It is not applicable to natural images—for, while increasing image size, the edges end up looking like straight lines. This property is not suited to natural images of leaves and flowers, and is only applicable to text and cartoons.

Multi-scale R-angle [27] descriptor, compared to all the other descriptors, is intrinsic to shape contours under translation, rotation and scaling, because the other methods need normalization for scaling.

3.4.3 Centroid Contour and Angle Code Descriptors

The Centroid Contour Descriptor (CCD) used by Sangle et al. [28] measures the distance between the center and the boundary points, and is invariant to translation and rotation. If a user knows the location of the starting point, the image produces the same shape signature for the rotated images. The Angle Code Descriptor (ACD) computes the continuous orientation angles of leaf shapes but provides limited shape information. So they combined both CCD and ACD to retrieve all the essential information of a leaf image and applied these methods to the mango, tulsi, rose and Asoka tree species. The CCD and ACD were used to extract, oblong and orbicular leaf shapes and to identify leaf species in [29]. Knight [30] developed android app for identifying 6 different classes of leaves. He used CCD and ACH for extracting leaf features. Thangirala [31] proposed CCD with Centroid Contour Gradient for broadleaf classification and used CCG to extract leaf gradients between two points on the leaf's contour. These points were used to measure the angles between the tip and the base. Bong et al. [32] suggested to normalize the tip and base of the leaf and used centroid contour gradient (CCG) to capture the curvature of the tip and base of the leaf. They achieved 99.47% classification accuracy by using feed-forward back propagation network as classifier Fotopoulou et al. [18] advised to convert the centroid contour distance and angle code sequence into 1D time delay sequence and he measured similarity of leaf shapes through Multidimensional Embedding Sequence Similarity (MESS).

3.4.4 Point and Edge-based Feature Descriptors

A new descriptor called the shape context was introduced in [34] to dissociate shape information from different shapes. It is a technique used to extract point information from a shape's contours, measure similarity differences between feature vectors of various points in an image, and isolate information from the neighboring pixels of an image. The transformation of an object does not affect shape context information. It is invariant to rotation since it performs log polar operations while computing shape context information. It is invariant to small affine transformations, occlusions, the presence of outliers, and is applicable to clear images. Shape context is used to calculate the local and spatial information of an image. In [35], an advanced shape context method was introduced to reduce computational cost. In this method they used two sets: a voting set and a computing set. While the voting set was used to build the histogram information of the shape, the computing set was used to compute the shape context information of various shapes. This method was used for polygonal shaped leaf images. The researchers of paper [36] proposed a new technique in shape context termed the Inner Distance Shape Context (IDSC), where the Euclidean distance is used to compute the cost matrix between two shapes. But it does not consider how many line segments are crossed in shape boundaries and, further, increases the computational cost. The technique solves the problem above by calculating the length of the shortest path with in shape boundary, and is invariant to articulation points that requires complex matching algorithm to compare a set of points. The inner distance shape context (IDSC) technique was proposed in [37] for articulated shape recognition and it is a very useful technique when the veins in leaves are damaged. The IDSC cannot store information on compound and serrated leaves or model the local details of leaf shapes well. It models only global information and misses some local information. Zhao et al. [38] introduced the Interdependent Inner Distance Shape Context (I-IDSC) to calculate the shape context with different aspects, but different plant species can have a common shape and the I-IDSC discriminates between leaves with similar shapes but different margins. It accurately classifies both simple and compound leaves, retains the most discriminative information, is very fast and offers cheap storage.

A Histogram of Curvature over Scale (HoCS) [39] is method to measure histogram features in one single point because it is simple to compute, compact and requires no alignment. It is a multi-scale invariant integral curvature measure calculated from circle-centered point. It gives natural notions of scale by resizing the image in segmented areas. It is robust to noise and invariant with rotation. It also removes holes in leaf images, extracts curvatures from boundaries, and measures smooth as well as serrated margins. This technique was used in the paper [40] to extract the arc and area features of lobe-shaped leaf margins, but it is not suitable for all leaves. This technique was also used for Costa Rican species. The HoCS, however, is not articulation invariant. An active shape model was proposed in [42] to find edge points and leaf tip points by overlapping two leaf points and tracing their continuous shape. The model was used for slender and thread-type leaves. An active polygonal model technique was used by Cerutti et al. in [42] to extract the tip and base information of a leaf by computing 10 feature points such as the base, base angle, tip of the angle and the isosceles triangle. This model fits polygons on images, helps to preserve corners, and extracts information on leaf tips and bases. Toothed leaf margins are represented as a string. This method presents information on leaves semantically, and is most useful, especially when the leaf is unavailable at a time. The drawback, however, is the danger of misclassification of the leaf margin when the margin in question is imperfect. Du et al. [43] presented a leaf species identification method using shape matching technique. They adopted Douglas–Peucker approximation algorithm to get the attributes of the leaves and proposed a modified dynamic programming (MDD) algorithm for shape matching. This method is suitable even if the leaves are overlapped, distorted and partial. It works with any number of dimensions and extracts a small number of points by splitting the entire contour into small curves. It depends on the starting point, and is a pure geometrical algorithm to obtain a smaller number of vertices. It also affects from noisy images.

3.4.5 Edges and Corner Points

Edges are significant features of leaf images in terms of measuring sharp variations in images. The Sobel edge detection operators were used to extract edge features from images in [44]. From the edges, feature points were found which intersect the edges and achieved 100% accuracy with 13 different plants. The model ascertains damages to veins. Corner features [45] are useful to find the similarity of leaf images because corners are intersections of two different edges or interest points under various different directions and lighting conditions. They are stable across different sequences, useful when there is damage to the corners, and are the same for all leaves. Harris Corner detectors are used to find the different directions of contours directly. The angles are arranged in ascending order, stored in an array and compared to find out the least angle of the unmatched image.

Tekkesinoglu et al. [46] used morphological transformation and edge detection techniques to identify the leaf boundary of overlapped (Hevea leaves) rubber tree leaves.

3.4.6 Leaf Tooth, Tip, Margin

A tooth is a depth incised towards the sinus and it is different from a lobe. In [47], the authors estimated the tooth's area, perimeter and internal angles for the whole tooth of Tilla trees by applying the tooth-finding algorithm. They found the points on edges by calculating the centroid distance from the center to the edge and thereafter marked the sinus of the margin. Each tooth can be represented as a triangle containing a tip and sinus on both sides. They used LDA to classify the species of Tila family such as Tilla platypyllus, Tilla Americana and Tiila Tomentosa, and achieved a classification accuracy of 68.3%. Susan corner detectors were applied to detect leaf image corners and Non-leaf image corners are removed using Pauta Criteria in [48]. The leaf number, leaf rate, leaf sharpness and leaf obliqueness of leaf tooth features are measured and the leaves are classified using the sparse representation of leaves. The tangential angle approach was used in [10] for finer angular details of the leaf boundary. Geometrical distances such as mid vein length, apical extension length, basal extension length and leaf length may be used to measure base angles and apex of different shapes of leaves.

3.4.7 Geometric Features

Geometric features are used for leaf classification because they are of low-dimensional compared to other features, incur low computational cost and take less time to extract the features. Morphological features were used for weed classification by Cho et al. [49]. Singh et al. [50] used 5 basic geometrical features and 12 digital morphometric features with fourier moments to classify 32 different plants. In [53] Dornbusch and Andrieu recommended the Lamina 2 Shape algorithm to analyze lamina-shaped gross leaves to measure their length, width and area. They estimated the accuracy of the width by calculating a predefined lamina shaped model. This algorithm forms equally-spaced perimeters on the area of the leaf and is not suitable for all types of leaves. The Waddle disk diameter method was used to measure the roundness of leaf area for grass-like species such as ryegrass, wheat and brome grass. Hossain and Amin [51] used biometric-based geometrical features of leaves for broad and flat leaves by selecting reference points from leaf blades and leaf bases. The researchers Wu et al. [56] proposed 5 geometric features—diameter, physiological length, physiological width, area, and perimeter—and 12 morphological features including smoothness, aspect ratio, form, rectangularity, narrowness, perimeter ratio of the physiological length and width, and 4 vein features of the leaf. These features were used to recognize 32 different kinds of plants.These features were tested on the Flavia dataset to classify the leaves. In [54] the authors applied digital morphological features to classify 32 different plant species and rate them. Geometrical and morphological features were used in [55] to classify compound leaves. Instead of extracting the features of the whole leaf image, the authors successfully extracted geometrical features from each leaflet of an image of clustered potato and tomato leaves. Kaif and Khan [56] used geometrical and shape-defining features such as the shape of the object, sets of horizontal and vertical lines, endpoints, boundary points, slopes between two lines and Fourier descriptors for the TRS invariant features. The authors of paper [57] used the morphological covariance method to extract coarseness, anisotropy, and textural data of images. They used structuring elements to represent the contour of curves, extracted edges from the leaf contour, and extracted shape information from images and introduced the Circular Covariance Histogram to extract venation information from leaf images using the circular structuring element. They divided the leaves into equal parts and calculated the statistical features separately, as both deformable and whole leaves have the same structure, so features extracted from one place are used as a vector for deformable objects. Dutta et al. [58] used geometrical and morphological characteristics of leaves to classify mango plants. Most researchers use geometric features for leaf classification, alongside weed detection because of the fewer dimensions involved, but they do not consider details of leaf margins. Leaf margins contain most of the details, and are only applicable to smoothed leaves. Manik et al. [20] used morphological features of Anthocephalus cadamba to identify diseases in leaves.

3.4.8 Texture of Leaf

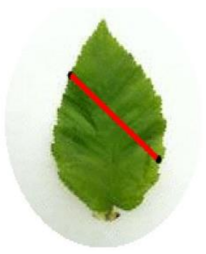
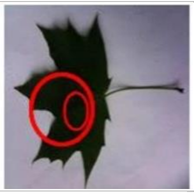
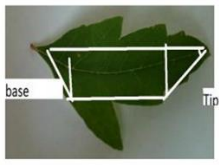
An image texture is recognized by a set of metrics designed to quantify the perceived texture of an image. It gives us information about the spatial arrangement of color or intensities in an image or a selected region of an image. Image textures, which can be artificially created or found in natural scenes captured in an image, can be used to classify images. A texture-based feature extraction method extracts the characterization of regions in a leaf image by means of its texture content such as smoothness, roughness or silkiness. The texture of leaves differs for the same species of leaf.

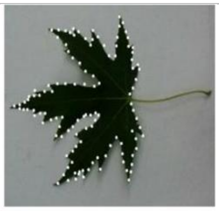
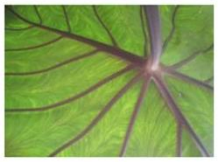
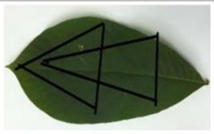
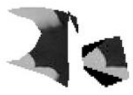
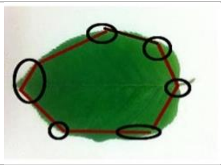
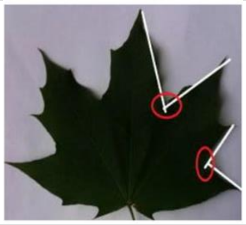
3.4.8.1 Texture Features Based on Fractals

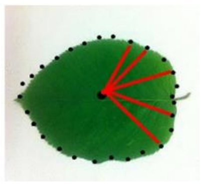
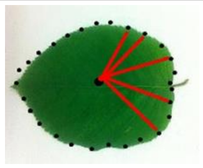
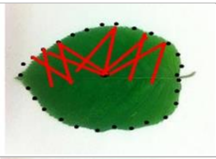
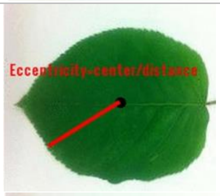
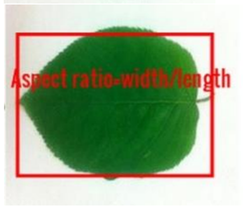
The topological structure is used to measure how close two objects are to each other. In [80], the authors used a Lie group of region structures to measure the texture of weeds and provide information about pixel intensity and spatial features of broadleaf weeds. The smooth manifolds of local symmetries were derived at by applying the Riemannian Manifold on the leaf surface. The dimension of a region covariance of the leaf surface is lower than that of the original image. It extracts multiple features such as information on edges and directions. Fractals measure the self similar texture of leaves as well as the roughness of the leaf surface. A multi-scale Minkowsi fractal dimension method was used to analyze and recognize leaf images in [61, 62]. This method extracts outline and vein features as curves. Usually, objects and patterns have distinct geometric natures in fractals and, in order to overcome this difficulty, they used the multi-scale Minkowsi fractal dimension technique for classifying Passiflora leaf morphometry. The researchers used new fractal refinement technique for classifying species based on contour, contour nerves, nervure fractals of three different levels. Mutchar and Fatichah [63] used lacunarity feature for leaf classification as the fractal dimension cannot discriminate between two objects with different patterns/texture. It measures the spatial distribution of gaps with certain image textures. It collects various image features and extracts energy signature from leaves. He evaluated 20 different classes of Brazilian flora using Linear Discriminant Analysis and achieved 86.00% of classification accuracy. Vijayalakshmi et al. [61] extracted texture using Gabor filter with 30-degree rotation angle in a 5 9 5 pixel neighborhood and obtained 13 different structural characteristics of a leaf compared to other kernel-based methods that use a 45-degree rotation angle to extract only 8 different statistical measures. But it does not obtain any invariant features. Singular value decomposition method was directly applied on a real matrix to classify texture characteristics with high-level factorization and provides good results in varying lighting conditions. A gray-tone spatial dependency matrix and LBP patterns were applied for the classification of medicinal leaves. The Local Gabor phase quantization (LGPQ) scheme proposed in [65] to extract different features of texture changes gradually along with a rich set of discriminated information because of the magnitude of information it carries. The authors extracted the entropy, mean, skewness, standard deviation and variance.

3.4.8.2 Local Binary Patterns Based on Texture

The Local binary pattern (LBP) is an image feature, which transforms image into an array of values. It describes about the changes in the neighboring pixels. Qi et al. [66] introduced a pair-wise rotation invariant co-occurrence local binary pattern (PRICoLBP) and applied to color images. It represents the local curvature as well as edge contour information. This technique was applied to various databases comprising flower and leaves. A LBP histogram Fourier feature (LBP-HF) identifies uniform patterns using Fourier descriptors. It stores all uniform patterns in a single bin and the authors used all the information on pixels, leaf interiors and exteriors separately. A modified local binary pattern where LBP binary values are calculated based on thresholding. It lends same LBP code for two different patterns. To overcome this problem the mean and standard deviation of the neighboring pixels were taken into account. It captures the structural relationship between the gray values of the pixels in the neighborhood. The LBP was combined with the gray-level co-occurrence matrix for tea leaf classification. In the basic LBP, every pixel needs to be calculated for obtaining LBP values, and computing the LBP is a time-consuming process. To circumvent the problem, the authors introduced a non-overlap window that includes a center pixel and its neighbor pixels in a single gray-level image. There is no overlap between the windows in this technique. Since the GLCM is used to calculate the relationship between two windows, it produces multiscale texture features. A multiscale local binary pattern was applied on the path integral (pi-LBP) in [66]. In all multiscale LBPs, local information is encoded individually in each scale, but the pi-LBP can effectively encode the cross-scale correlation and provides better texture description. A pixel-based LBP was used instead of computing global information built on a block-based LBP the authors computed LBP based on center pixel of a half-size window which determines how much local and global information is included in the texture descriptor. It produces powerful relations for the intra-class variability of textures. Sumathi et al. [66] used Gabor filter for textural, statistical and spatial frequency domain relationships in leaf classification. The LBP variance [40] was applied to classify Costa Rican plant species. It detects micro texture veins as well as areas between veins and reflections. It returns a histogram of features and counts the position in which it corresponds to the particular leaf texture which has an LBP code. Siricharoen et al. [37] used 13 textural features and 6 different Tamura’s texture features for plant disease monitoring in a mobile cloud environment. The shortest path texture context [36] measures the shortest path along different orientations. Combining texture information and global shape information with local patches, the authors used gradient changes for lighting invariance.

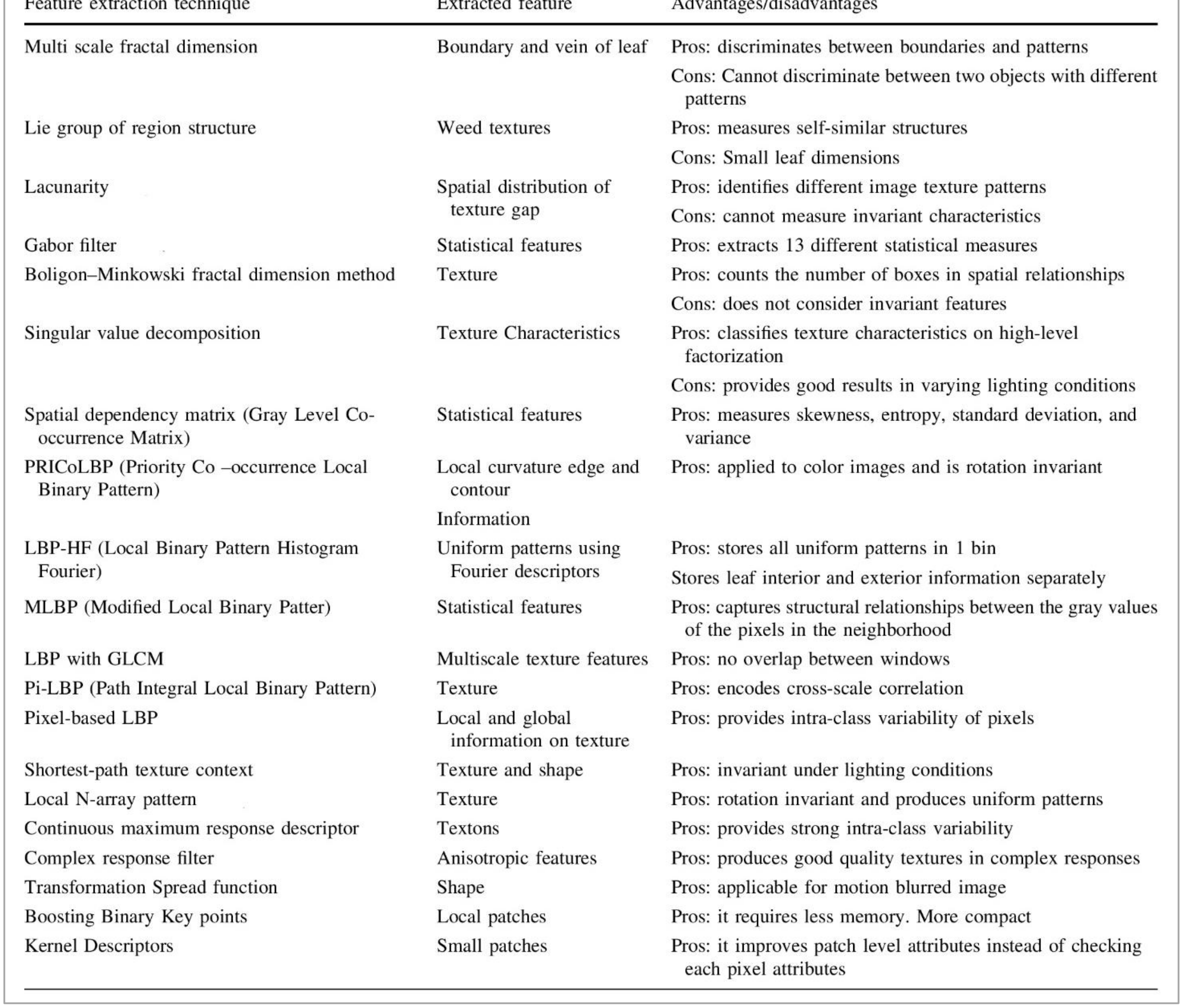
Feature extraction technique	**Image**	**Extracted feature from leaf**	**Pros and cons of feature extraction technique**
Shape Context		Point information from shape contours	Isolates information from nearby pixels and is invariant to affine transformation, occlusions and the presence of outliers, Applicable to only unaffected images
Advanced shape context		Relations between Salient and margin points	Reduces computational costs, Applicable to polygonal objects
Shortest-path texture context Inner-distance shape context		Leaf vein	Measures the relative orientation along the shortest path Used for texture non uniform illumination changes of leaf veins Useful when veins are damaged and models only global information Cannot store information on compound and serrated leaves.
Histogram of curvature over scale(HOCS)		Histogram information in one single point	Robust to noise and rotation invariant Only suitable for lobe shaped leaves It is not articulation invariant
Douglas Peucker contour Approximation		Leaf shape	Smooth contour obtained with small number of vertices. It is varying in translation, rotation and scaling.
Contour characteristics points		Contour points selected depends on the curvature of contours	It is robust to translation, rotation and scale invariant.
Active shape model		Leaf tip	Finds leaf tip points and overlapping leaf tips. Used only for slender and thread type leaves.

Feature extraction technique	**Image**	**Extracted feature from leaf**	**Pros and cons of feature extraction technique**
Active polygonal model		Leaf tip and leaf base	Preserves leaf corners Leaf tips vary in images, And there is damage to leaf corners.
Contours of string		Leaf margin	Semantically represents leaf margins. Leads to misclassification when there are imperfect leaf margins and overlapped leaves.
Curvature scale space		Leaf venation	Finds the starting and ending points of leaves. Produces noise and is sensitive to rotation.
Multi scale triangular area representation		Concavity and convexity of images	Affine invariant and robust to noise. Not scale invariant.
Multi scale arch height descriptors	1-scale 2-scale and 4-scale 	Leaf margin	Measures the arch height of lobe shaped and palmate shaped leaves. Unsuitable for overlapped leaves. Normalization applied for scaling and rotation, taking up time.
Multi scale bending energy		Energy	Sensitive to local features of leaf shape contours. Provides low immunity.
Curvlet transform		Curvelet features	Useful for small objects. Unsuitable for natural images.
Multi scale R-angle descriptor		Leaf margin	Intrinsic to shape contour under, Translation. Rotation and scaling. No need for normalization.

Feature extraction technique	**Image**	**Extracted feature from leaf**	**Pros and cons of feature extraction technique**
Centroid contour distance, Angle code histogram.		Contour points and orientation angles.	Invariant to translation and rotation. Used for compound, oblong and orbicular leaf shapes. Applicable only to leaf tip and base.
Contour Key points		Contour Key points are extracted and represented as histogram bins by using fuzzy score	Solves intra class problem of same species.
Complex network Descriptor		Measures degree and joint degree of leaf boundary.	Invariant to scaling and rotation. Noise tolerant.
Geometric features	 	Eccentricity Aspect ratio Leaf area Leaf perimeter Major and minor axis Solidity	Semantically represents leaf margins. Leads to misclassification when there are imperfect leaf margins and overlapped leaves.

3.4.9 Textons

Textons are used to construct texton dictionaries created based on filter responses in spatial and frequency domains. For rotation invariant databases, the authors of [58] constructed a continuous maximum response descriptor to distinguish between and intra-class variations and a principal curvature descriptor for strong intra-class grouping ability. These techniques are useful for leaf databases with both interclass and intra-class variations. Minu and Thyagarajan [73] used texton with MPEG 7 visual features to recognize flower images. They also presented an ontology based image retrieval system for asteroideae flower domain in their paper [68]. Guo et al. [65] classified rotation invariant texture by first finding out dominant orientation and then extracting anisotropic features by this orientation. They also proposed two statistical texton based methods to validate their approach. Anisotropic images change in appearance and rotate to produce good quality textures. The average and standard deviations of responses were computed in 8 different directions and a joint sort was used to find the local patch. These methods can be used to classify leaves in rotation invariant leaf databases. Table 5 shows some of feature descriptors used in leaf recognition.



Feature extraction technique	Extracted feature	Advantages/disadvantages
Multi scale fractal dimension	Boundary and vein of leaf	Pros: discriminates between boundaries and patterns Cons: Cannot discriminate between two objects with different patterns
Lie group of region structure	Weed textures	Pros: measures self-similar structures Cons: Small leaf dimensions
Lacunarity	Spatial distribution of texture gap	Pros: identifies different image texture patterns Cons: cannot measure invariant characteristics
Gabor filter	Statistical features	Pros: extracts 13 different statistical measures
Boligon–Minkowski fractal dimension method	Texture	Pros: counts the number of boxes in spatial relationships Cons: does not consider invariant features
Singular value decomposition	Texture Characteristics	Pros: classifies texture characteristics on high-level factorization Cons: provides good results in varying lighting conditions
Spatial dependency matrix (Gray Level Co-occurrence Matrix)	Statistical features	Pros: measures skewness, entropy, standard deviation, and variance
PRICoLBP (Priority Co –occurrence Local Binary Pattern)	Local curvature edge and contour Information	Pros: applied to color images and is rotation invariant
LBP-HF (Local Binary Pattern Histogram Fourier)	Uniform patterns using Fourier descriptors	Pros: stores all uniform patterns in 1 bin Stores leaf interior and exterior information separately
MLBP (Modified Local Binary Patter)	Statistical features	Pros: captures structural relationships between the gray values of the pixels in the neighborhood
LBP with GLCM	Multiscale texture features	Pros: no overlap between windows
Pi-LBP (Path Integral Local Binary Pattern)	Texture	Pros: encodes cross-scale correlation
Pixel-based LBP	Local and global information on texture	Pros: provides intra-class variability of pixels
Shortest-path texture context	Texture and shape	Pros: invariant under lighting conditions
Local N-array pattern	Texture	Pros: rotation invariant and produces uniform patterns
Continuous maximum response descriptor	Textons	Pros: provides strong intra-class variability
Complex response filter	Anisotropic features	Pros: produces good quality textures in complex responses
Transformation Spread function	Shape	Pros: applicable for motion blurred image
Boosting Binary Key points	Local patches	Pros: it requires less memory. More compact
Kernel Descriptors	Small patches	Pros: it improves patch level attributes instead of checking each pixel attributes

Table 5: A summarization of texture, texton and LBP descriptors

3.5 State-of-the-Art Classification Techniques

Plant species classification can be carried out by botanists easily, but computer-assisted systems cannot do so as easily. Consequently, plants are classified through leaf shape, vein, color and the texture of the leaves. Plant species are classified through different classifiers. A classifier requires two sets of data, a training set and a test set, but does not consider class relationships and the illumination invariance and positional invariance of images. Certain authors use manifold learning for classification since it preserves local neighborhood structure, and highdimensional data is mapped into a low-dimensional structure. It also considers all illumination and positional challenges and processes noisy images. Compared to linear and supervised classifiers, manifold learning offers a good accuracy on plant species identification (Fig. 4)

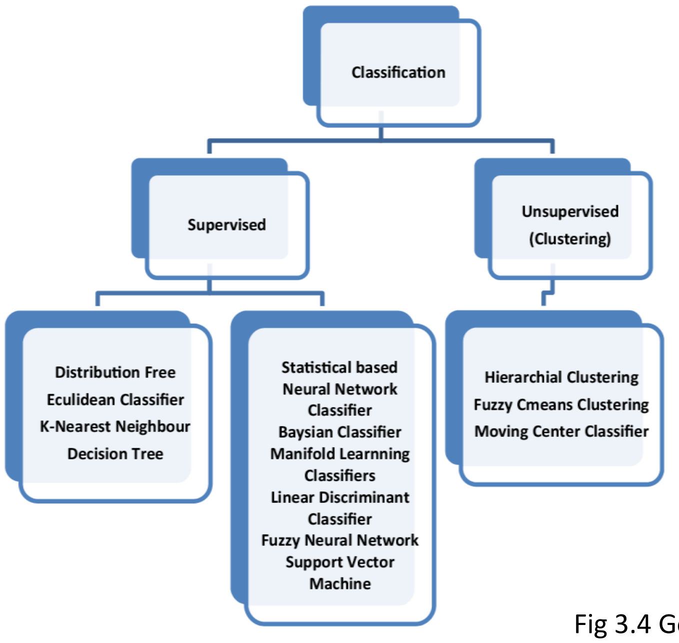


Fig 3.4 General classification technique

3.5.1 Artificial Neural Networks

A neural network is a machine learning technique used for classification. The authors of the paper [124] identified disease in cotton, lemon and orange with the color feature and achieved 76.41% abnormality and 9.09% abnormality in leaf disease detection. Back propagation neural network (BPN) was used to classify half leaves based on the boundary tokens of shapes such as the angles and sinus of leaves in [125]. The authors examined 111 leaves of 14 different classes. It is a feed forward, self-adaptive network. Weights are adjusted based on the minimum mean square error. It takes longer time to train the network. Bagalkote et al. [126] used the BPN to classify grape varieties using texture and wavelet features and achieved 93.3% accuracy. Anami et al. [127] used neural network to identify affected species of leaves based on color and texture features and identified 85% of affected vegetables and 80% of normal ones. Neural network was applied in [128] for plant disease classification and identification, based on the color co-occurrence texture features of the leaf. The BPN was used in [129] with the edge features for classification of leaves such as the neem, pine and oak and achieved 90.45% classification accuracy. The authors of the paper [130] used the BPN to classify the night jasmine, arka (blue madar), mango, neem, and shigru (moringa/drumstick) and achieved 85% accuracy. The radial basis function is a three-layer feed forward network used for image classification, and produces faster training speeds compared to the Multilayer Perceptron (MLP). Sumathi et al. [98] used this approach to classify 90 samples and achieved 85.93% accuracy with a minimum mean square error. This method works well on spherical and regularized linear spaces. Akif and Khan [65] used the ANN to classify 817 samples of 14 different trees with morphological features, utilizing the Fourier descriptor and shape-defining features and achieved 96% accuracy. The author used the ANN to classify 80 leaf images with 10 different classes with 7 different morphological features and achieved 98.8% classification accuracy. The authors of paper [131] accounted the single hidden layer feed forward network for classification. There is no need to use a kernel function to approximate the weights, given that it updates the weights randomly for fixed bias inputs. It has no control parameters such as learning rates, learning epochs and stopping criteria. They achieved 98.17% classification accuracy. Chaki et al. [26] designed a neuro-fuzzy system with a back propagation multilayered feed forward network to classify 930 images of 31 classes and achieved 97.6% accuracy. Because neuro-fuzzy system uses the probability of classes, to avoid problems in the ANN, fuzzy C-means clustering works by assigning each membership to each data point corresponding to other data points which belong to more than one cluster and it gives good results for overlapped datasets. In k-means clustering, data points belong to more than one cluster center, but here they are assigned. The authors of paper [42] used this algorithm to classify species of plant databases with the specified margin structure. Balasubramanian et al. [64] formulated the fuzzy relevance vector machine to classify 60 categories of leaf images with shape and texture features. This method helps to select the optimum features of an image, achieving 99.87% accuracy. Sharma and Gupta [89] developed a system to classify agriculture and Ayurvedic plants using a multilayer feed-forward network with back propagation algorithm. They tested their system with 440 leaves of 16 classes and obtained classification accuracy greater than 90%.

3.5.2 K-Nearest Neighbours

The K-Nearest Neighbor (K-NN) is a simple technique used to classify objects with the closest training samples in feature spaces. Images are classified, based on the majority voting of its neighbors. Du et al. [26] used the KNN for classifying plants of 30 species with 2422 image samples based on edge, vein and ring projection fractal wavelet features as a new shape feature, and achieved 87.14% of classification accuracy with the size of the feature vector as 20. The researcher in [79] used the KNN classifier to classify 100 leaf species with 1600 samples using the leaf margin as a feature and an interior texture histogram of 64 different feature vector images, and achieved 75.5% of classification accuracy. Jose et al. [40] used the KNN for Costa Rican plant identification in the Flavia dataset using the features of the 0.5 HoCS (Histogram curvature Scale Space), LBPV (Local Binary pattern Variance), R1P8 (1 rotation with 8 pair of neighborhood pixels), and R3P16 (3 rotations with 16 pairs of neighborhood pixels) with k = 10 and achieved an accuracy of 99.1%. The authors in [54] used KNN with fractal dimension of the RPWFF to classify a total of 2422 images of 30 different species and achieved 87.14% of classification accuracy. Zhao et al. [37] used KNN to classify the Swedish, ICL, Smithsonian and Plummers Island datasets with a pattern counting approach and achieved 97.07, 73.08, and 72.28% classification accuracy respectively. Arunpriya et al. [89] experimented with fuzzy inference system, radial basis function network and K-nearest neighbour classifiers and classified tea species using leaf images and came to a conclusion that fuzzy inference system obtained better accuracy and took less time for execution compared to other two classifications.

3.5.3 Moving Center Hyper spheres

A Moving Center Hyper sphere classifier (MCH) [85] was proposed for high-dimensional features. In the KNN and neural network, the classification of plants is a laborious and space-consuming process. In the MCH, however, the features are arranged as n-hyper spheres. Using this classifier, 1200 leaf samples of 20 classes were tested with 23 moment invariant features and achieved 92.6% accuracy.

3.5.4 Bayesian Classifiers

The Bayesian classifier, a simple probabilistic classifier based on Bayes' theorem, compute the posterior probability for the targeted output. The researchers in paper [1] used the Bayesian classifier with the Fourier descriptor feature to classify 100 different kinds of leaves and achieved 88% accuracy. They used the linear classifier with the features of the polar Fourier transform, color, vein and 20 features of lancularity, solidity and convexity of shape for classifying the Flavia and Foliage databases and achieved 95.94 and 93.25% accuracy respectively.

3.5.5 Support Vector Machine

The support vector machine (SVM) is a linear classifier. The process of classifying leaf species calls for a multiclass classifier, because multiple leaf species are identified by multiclass SVMs. Compared to the neural network classifier, it performs better because of its selection of kernels. No prior training is called for, though it involves a huge number of images. In [38] the authors used SVM-RBF (Radial Basis Function) kernel to classify leaves of the Leaf snap database. The RBF kernels automatically produce a number of support vectors, centers, and weights during the training. A multiclass SVM [93] was applied on the Australian Federal dataset, Flavia, Foliage, Swedish and Middle European datasets with the texture features and Fourier transform descriptors and combined the features of interior and boundary descriptors extracted, and achieved classification accuracy of 100% in the AFF, 99.7% in Flavia, 99.8% in Foliage, and 99.2% in MEW (Middle European Woody) datasets. The authors of papers [111] used one versus all SVMs in the Flavia dataset with kernel level descriptors [110][111] and achieved an average accuracy of 97.5% (1585 training images of 32 species and 320 testing images), and 58%

with Image CLEF 2013 (7525 training images of 70 species with 1250 testing images. SVM classifier with the fractal dimension of the leaf shape with its lancularity features was used in [84] to classify 626 images of the Flavia dataset with an average accuracy of 95.048%. The SVM used to classify the Flavia and Swedish datasets with the features of the HOG and Zernike moments with 40 samples provided an average accuracy of 97.18 and 98.13% respectively.

3.5.6 Principal Component Analysis

The PCA is an algebraic technique used to select important correlated variables from images. Glozarian and Frick [54] used the PCA to classify species of different grasses such as wheat, rye and brome grass by extracting the shape, color and texture features of images and achieved 88 and 85% for Wheat and brome classification accuracy. The PCA with textural features extracted by the gray-level cooccurrence method was used to classify 390 leaves with 13 different kinds of plants and achieved 98% accuracy in [72]. The authors also extracted shape, texture, and color features from leaf images in [48] and optimized i.e. selected a subset of features using genetic algorithm and Kernel based Principal Component Analysis (KPCA) to improve the accuracy of classification.

3.5.7 Random Forest

An ensemble classifier, the random forest is used to construct a large set of trees at random. It runs efficiently on large databases, handles a large number of input variables without variable deletion, effectively estimates missing data, and maintains the accuracy of a classifier. It gives proximities between pairs of classes and, further, estimates crucial features automatically. During multiclass classification, if some data are missed, it leads to an imbalance in the data concerned. To resolve this problem, a direct ensemble classifier [99][98] is used for an imbalanced multiclass learning classifier. It is a combination of the 1-nearest neighbor and Naive Bayes or the K-nearest neighbor and Naive Bayes classifiers.

3.5.8 Convolutional Neural Networks

All classifiers handle a small number of images, except the CNN (convolutional neural network), which handles large set of images. For all classifiers, feature extraction is a separate space, since they cannot directly extract features from images. The CNN, however, extracts features directly from the images in question, disregarding illumination, lighting, shadowing or skewness. It is not rotation invariant but translation invariant, needing similar-sized images for classification. The CNN was used to classify large sets of images by Dyrmann et al. [99] and they trained 10,413 images of 22 species, achieving a classification accuracy of 86.2%. The authors of the paper [100] used the CNN to identify 13 different plant diseases and achieved 96.3% accuracy. The researchers of the paper [18] applied the CNN to identify legume species of soya bean, white bean and red bean using the vein morphological features of 422 images of soybeans, 272 red beans and 172 white beans leaves and achieved an average recognition accuracy of 96.9%.

Chapter 4

Results and Analysis

4.1 Dataset

All the datasets that has been used in this work are described in this chapter. We have worked with Swedish Leaf dataset and the results and analysis has been made. The dataset is described in the following section.

4.1.1 Dataset Description

We took Swedish Leaf Dataset [] in our experiment employing Convolution Neural Network Model.

- There are 15 different categories of leaf present in the database.
- Fig. 4.1 depicts a snap shot of one image from each category.
- Each category has 75 instances.
- Total 15*75 color-image with different resolutions

Table 4.2, Table 4.3 and Table 4.4 are 3 tables that describes 9 sampled leaves of each class 1, class 2 and class 3. In these tables, the 1st column denotes the name of the image, the second column shows the class label, 3rd column describes trace of train-test-validation split and 4th column describes the size of the original image.

4.1.2 Preprocessing

- Each leaf image is rescaled such a way so that higher spatial dimension between the length and width of the leaf image becomes 128.
- Then padding of the lower spatial dimension with border pixel values is done so that lower spatial dimension becomes 128.
- Resultant image is of size 128*128 color image
- Each class of data is divided into train, valid and test sets. For each such set, we select images randomly from each classes. In Table 4.2, Table 4.3 and Table 4.4 the trace of train-test-validation split is put in the 3rd column.
- We took 40% data as train data, 30% data as validation data and 30% data as test data.

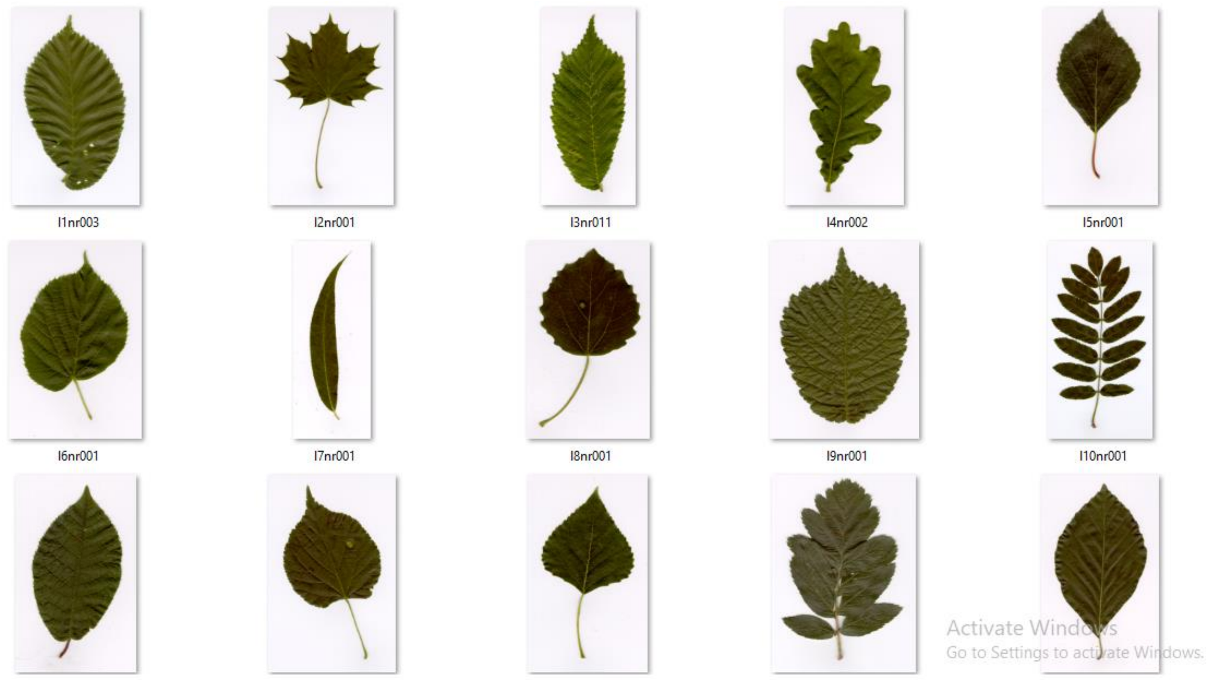


Fig 4. 1 Snapshot of leaf image database; one image from each category

# Name	Label	train =1, validation= 0, test=-1	shape: original	new size
l1nr001	1	0	(2508- 1423- 3)	128-128
l1nr002	1	1	(2143- 1147- 3)	128-128
l1nr003	1	1	(1994- 1294- 3)	128-128
l1nr004	1	1	(2482- 1457- 3)	128-128
l1nr005	1	-1	(1580- 917- 3)	128-128
l1nr006	1	1	(1154- 779- 3)	128-128
l1nr007	1	-1	(1304- 791- 3)	128-128
l1nr008	1	1	(1379- 804- 3)	128-128
l1nr009	1	0	(2207- 1508- 3)	128-128

Table 4. 2 describes leaf data of class 1

# Name	Label	train =1, validation= 0, test=-1	shape: original	new size
l2nr001	2	1	(2469- 1557- 3)	128-128
l2nr002	2	1	(3848- 2550- 3)	128-128
l2nr003	2	-1	(2947- 2374- 3)	128-128
l2nr004	2	0	(2445- 1721- 3)	128-128
l2nr005	2	-1	(3987- 2324- 3)	128-128
l2nr006	2	0	(3598- 2160- 3)	128-128
l2nr007	2	0	(2056- 1307- 3)	128-128
l2nr008	2	-1	(1994- 1558- 3)	128-128
l2nr009	2	1	(2909- 1771- 3)	128-128

Table 4. 3 describes leaf data of class 2

# Name	Label	train =1, validation= 0, test=-1	shape: original	new size
l3nr001	3	1	(2018- 992- 3)	128-128
l3nr002	3	-1	(2307- 1307- 3)	128-128
l3nr003	3	1	(2006- 1093- 3)	128-128
l3nr004	3	0	(1756- 842- 3)	128-128
l3nr005	3	1	(1743- 921- 3)	128-128
l3nr006	3	1	(2031- 1260- 3)	128-128
l3nr007	3	1	(1717- 1030- 3)	128-128
l3nr008	3	0	(1680- 1018- 3)	128-128
l3nr009	3	0	(1655- 854- 3)	128-128

Table 4. 4 describes leaf data of class 3

4.2 Different Models We Tested

In each model, there is 4 convolution layers for feature extraction and two fully connected layer for classification. We differ the number kernel in some model and activation function. We add Batch Normalization Layer in some network. To observe our model is over fitting or not we trace accuracy and loss value validation dataset for each epoch. We train network with mini-batch of size 32 and total 450 training data for 100 epoch.

4.3 Results and Analysis

In the following section different architectures of different models has been described and all the results, figures and their analysis has been made.

- In first model we did not use any activation after convolution operation. We added nonlinear activation only in full connected layers.
- In second model we added nonlinear activation relu after convolution operation.
- In third model we changed the nonlinear function of second layer by elu activation
- In forth model we added batch normalization layer after each convolution operation and other configuration is same as forth model
- In fifth model we used batch normalization after convolution operation then Leaky-ReLU layer.

4.3.1 Architecture of the First model:

Layer types	Output Shape	Parameter Number	Activation
input_1 (Input Layer)	(None, 128, 128 = 3)	0	
conv2d	(None, 128, 128, 64)	1792	f(x) = x
max_pooling2d	(None, 64, 64, 64)	0	
conv2d	(None, 64, 64, 32)	18464	f(x) = x
max_pooling2d	(None, 32, 32, 32)	0	
conv2d_2 (Conv2D)	(None, 32, 32, 16)	4624	f(x) = x

max_pooling2d	(None, 16, 16, 16)	0	
conv2d_3 (Conv2D)	(None, 16, 16, 8)	1160	f(x) = x
max_pooling2d	(None, 8, 8, 8)	0	
flatten	(None, 512)	0	
dense	(None, 32)	16416	tanh
dense	(None, 15)	495	softmax

==

Total parameters: 42,951

Trainable parameters: 42,951

Non-trainable parameters: 0

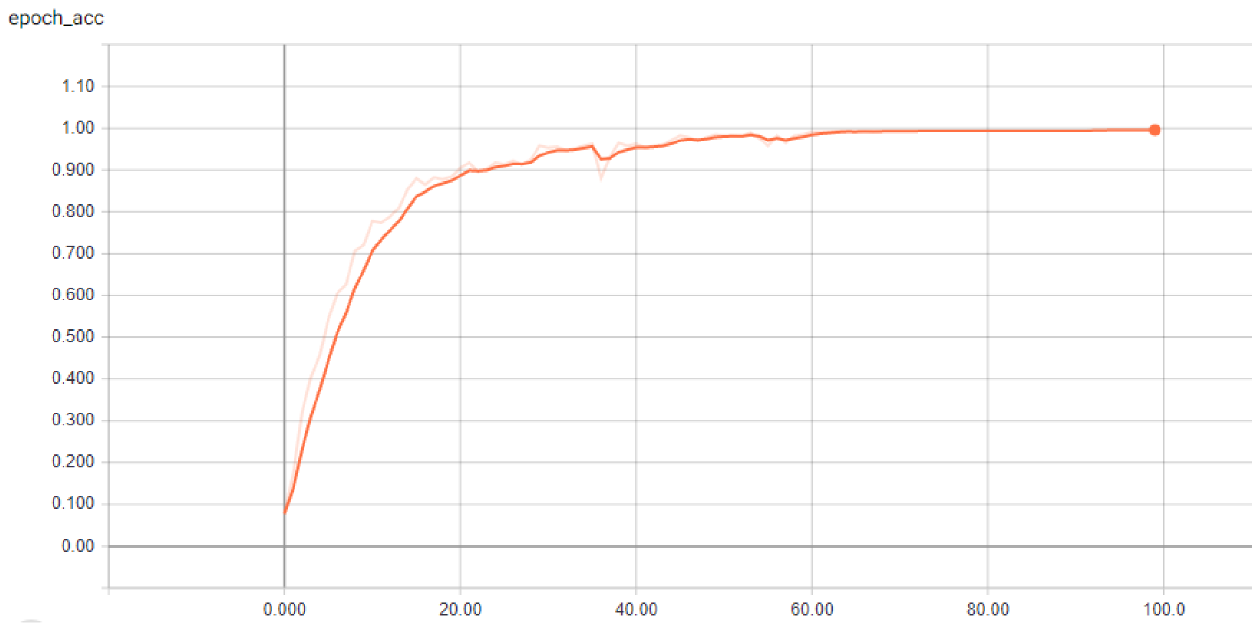


Fig 4. 5: Graph of training accuracy of 1st model. Horizontal axis describe number of epoch and vertical axis describe training accuracy. Training accuracy 1 means 100% accuracy 0.4 means 40% accuracy

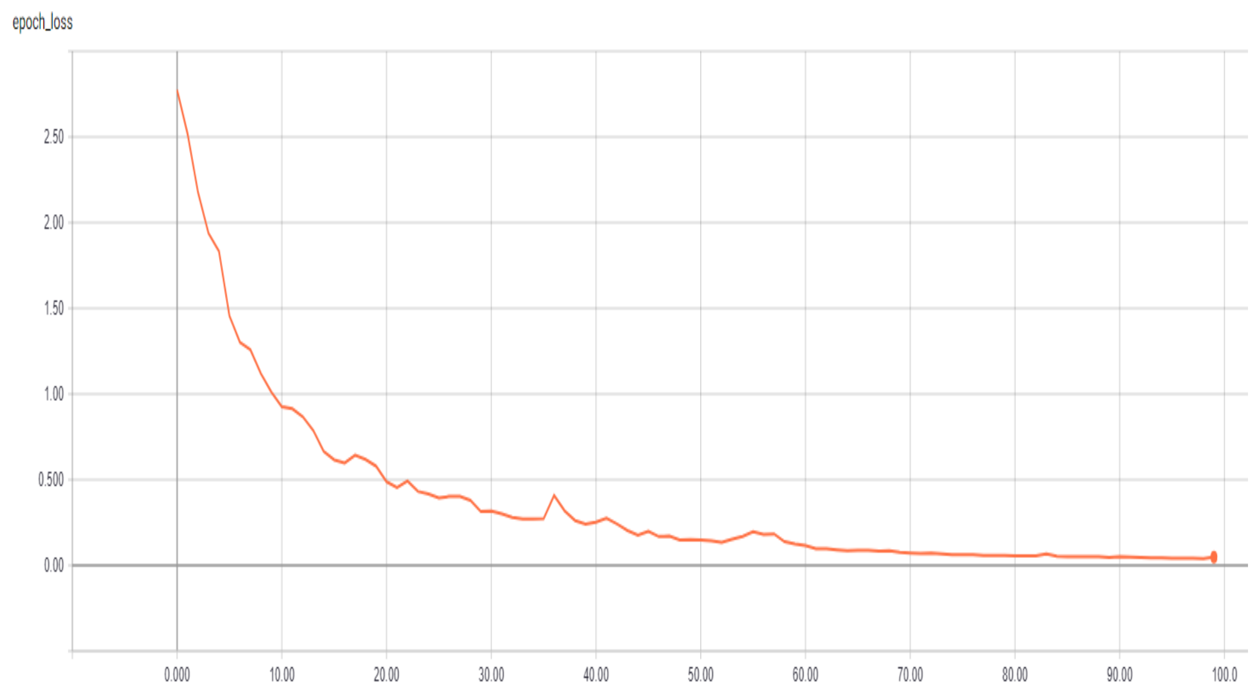


Fig 4. 6: Graph of training loss of 1st model. Horizontal axis describe number of epoch and vertical axis describe training softmax loss value.

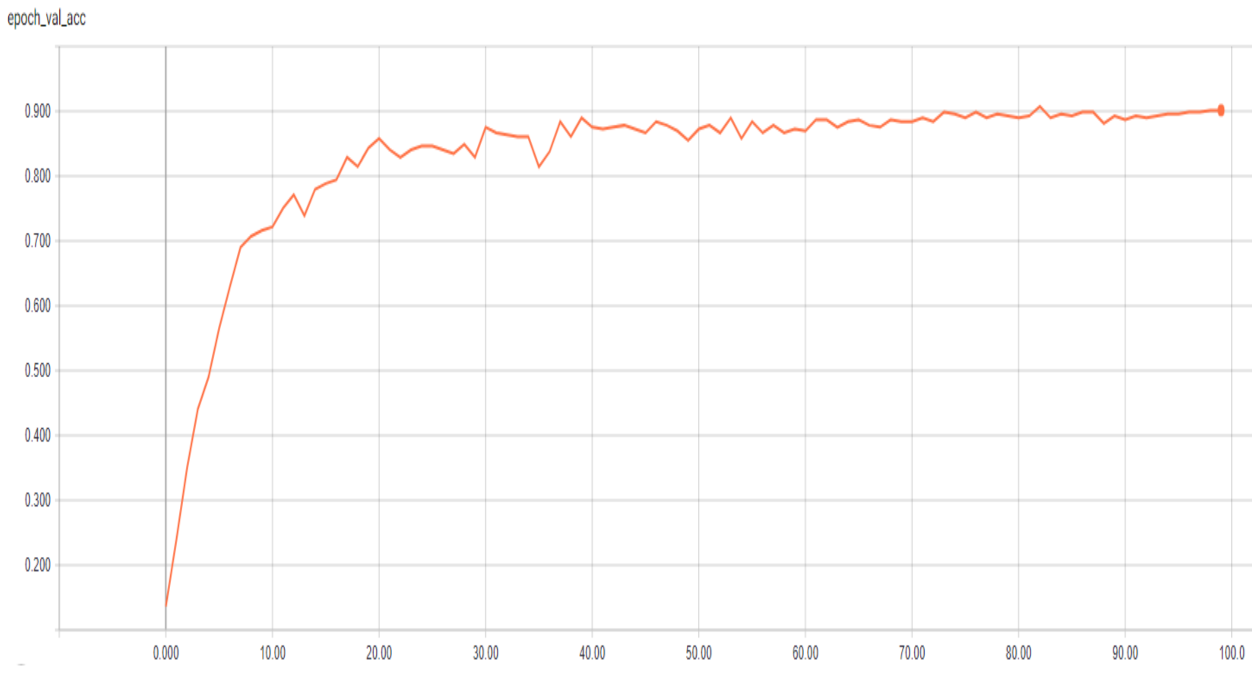


Fig. 4. 7 Graph of training accuracy of 1st model. Horizontal axis describe number of epoch and vertical axis describe training accuracy. Training accuracy 1 means 100% accuracy 0.4 means 40% accuracy.

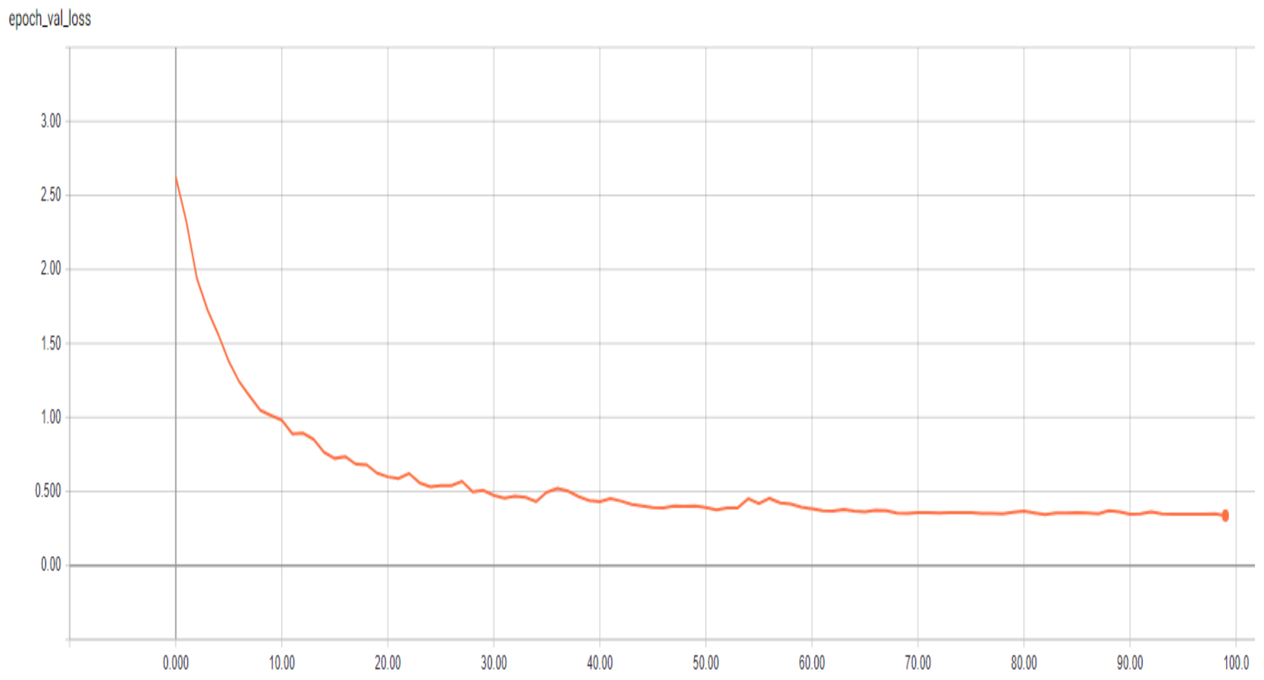


Fig 4. 8: Graph of validation loss of 1st model. Horizontal axis describe number of epoch and vertical axis describe training softmax loss value.

4.3.2 Architecture of the Second Model:

Layer (type)	Output Shape	# of Parameters	Activation
Input (Input Layer)	(None, 128, 128, 3)	0	
conv2d	(None, 128, 128, 64)	1792	ReLU
max_pooling2d	(None, 64, 64, 64)	0	
conv2d	(None, 64, 64, 32)	18464	ReLU
max_pooling2d	(None, 32, 32, 32)	0	
conv2d	(None, 32, 32, 16)	4624	ReLU
max_pooling2d	(None, 16, 16, 16)	0	
conv2d	(None, 16, 16, 8)	1160	ReLU
max_pooling2d	(None, 8, 8, 8)	0	
Flatten (Flatten)	(None, 512)	0	
dense (Dense)	(None, 32)	16416	tanh

dense (Dense)	(None, 15)	495	Softmax

Total parameters: 42,951

Trainable parameters: 42,951

Non-trainable parameters: 0

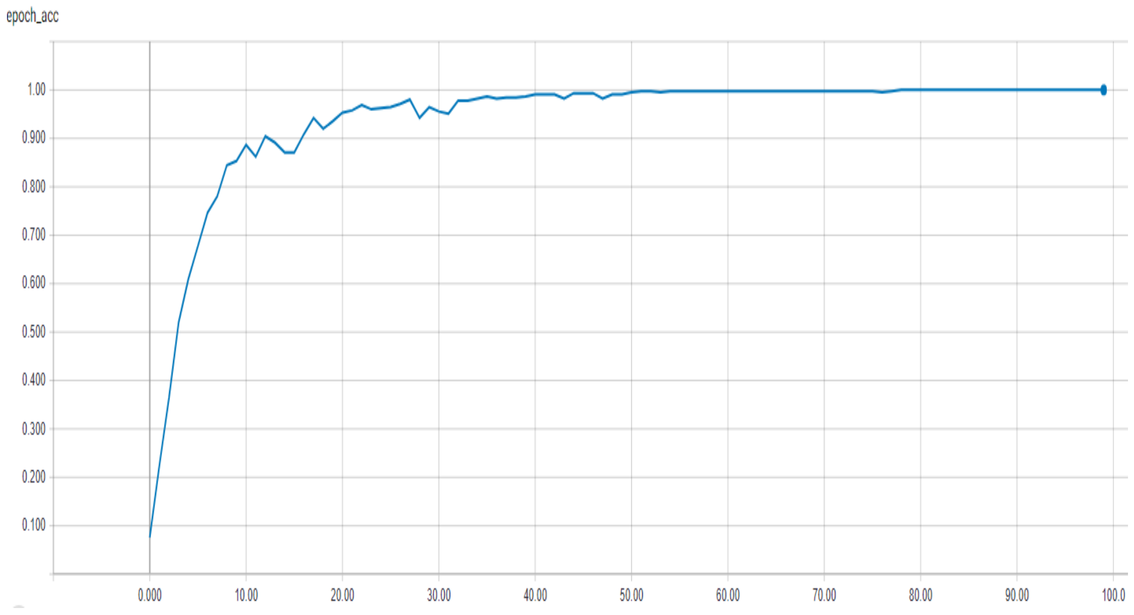


Fig 4.9: Graph of training accuracy of 2nd model. Horizontal axis describe number of epoch and vertical axis describe training accuracy. Training accuracy 1 means 100% accuracy 0.4 means 40% accuracy.

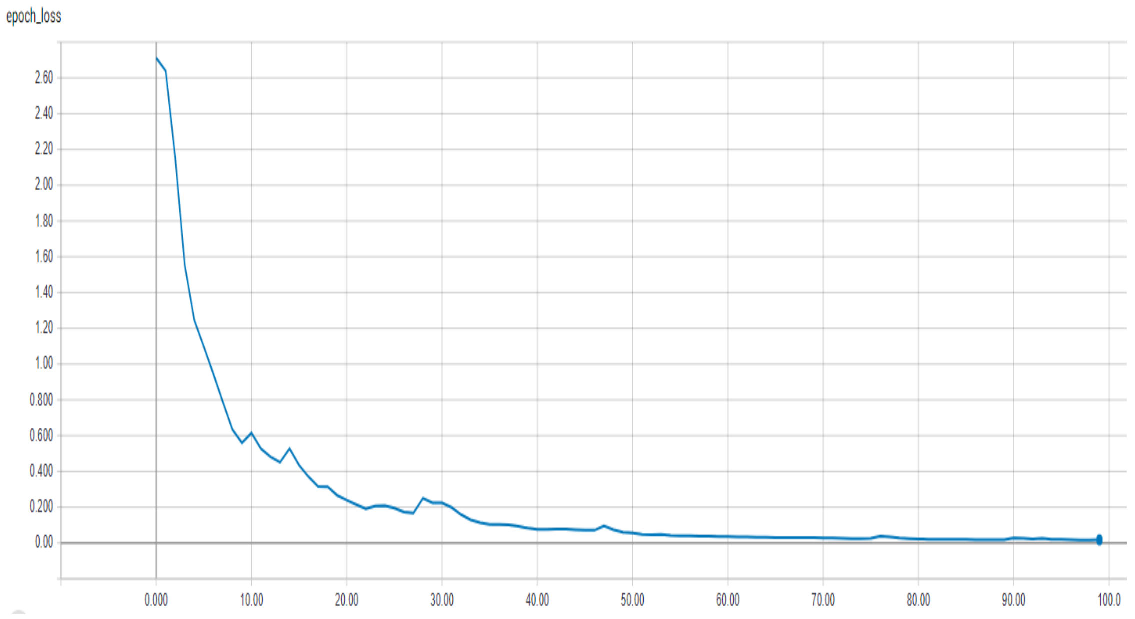


Fig 4. 8: Graph of training loss of 2nd model. Horizontal axis describe number of epoch and vertical axis describe softmax loss value.

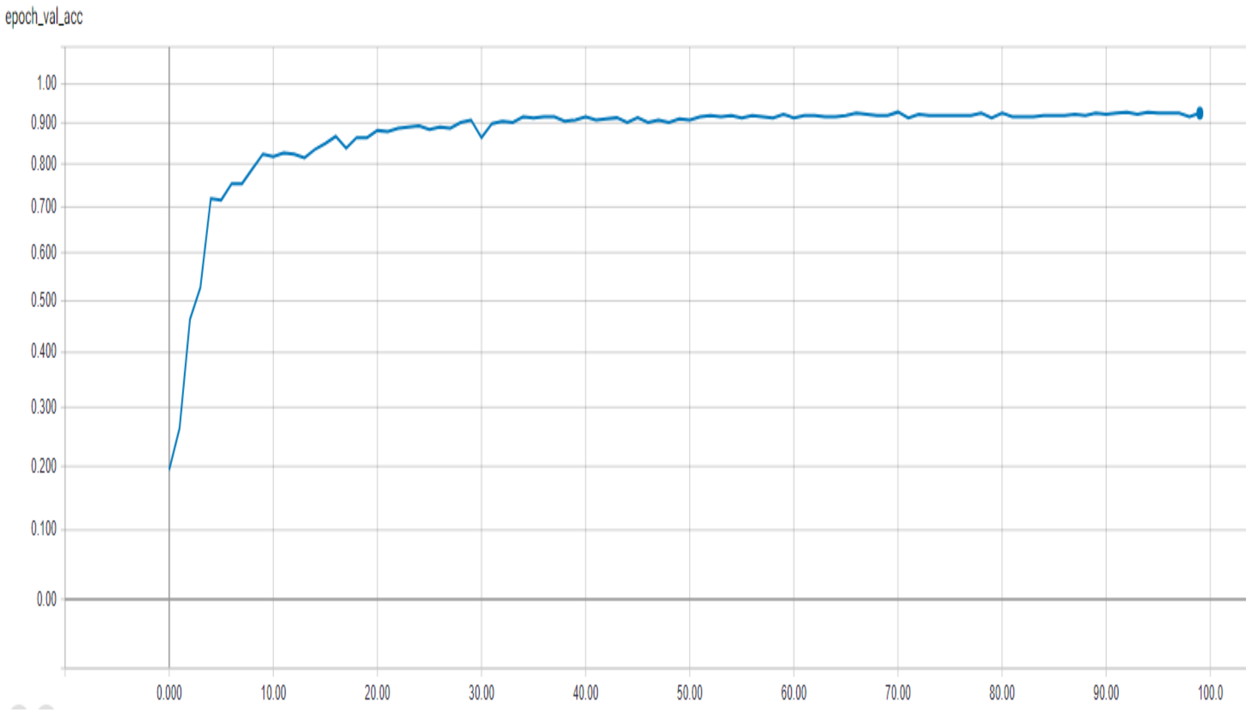


Fig 4.10: Graph of training accuracy of 2st model deprecate a graph. Horizontal axis describe number of epoch and vertical axis describe training accuracy. Training accuracy 1 means 100% accuracy 0.4 means 40% accuracy.

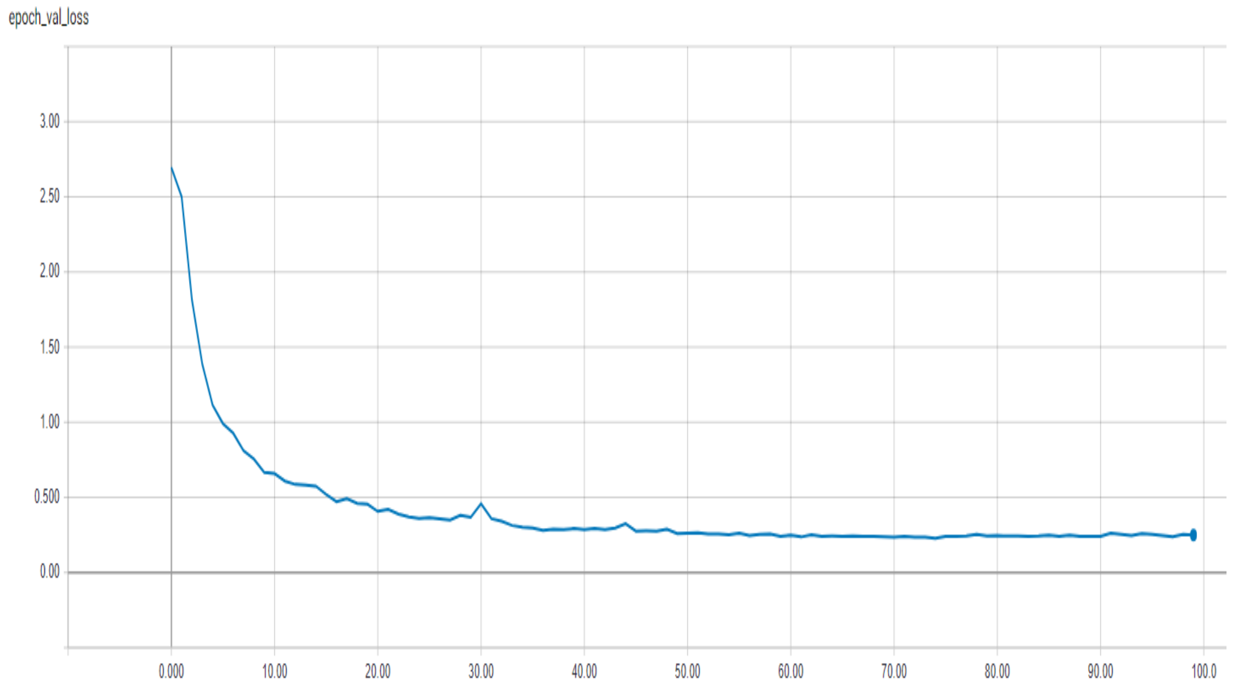


Fig 4. 9: Graph of validation loss of 2nd model. Horizontal axis describe number of epoch and vertical axis describe softmax loss value.

4.3.3 Architecture of the Third Model:

Layer (type)	Output Shape	Parameter #	Activation
input_1 (InputLayer)	(None, 128, 128, 3)	0	
conv2d (Conv2D)	(None, 128, 128, 64)	1792	elu
max_pooling2d (MaxPooling2D)	(None, 64, 64, 64)	0	
conv2d_1 (Conv2D)	(None, 64, 64, 32)	18464	elu
max_pooling2d_1(MaxPooling2)	(None, 32, 32, 32)	0	
conv2d_2 (Conv2D)	(None, 32, 32, 16)	4624	elu
max_pooling2d_2 (MaxPooling2)	(None, 16, 16, 16)	0	

conv2d_3 (Conv2D)	(None, 16, 16, 8)	1160	elu
max_pooling2d_3 (MaxPooling2)	(None, 8, 8, 8)	0	
flatten (Flatten)	(None, 512)	0	
dense (Dense)	(None, 32)	16416	tanh
dense_1 (Dense)	(None, 15)	495	softmax

===

Total parameters: 42,951

Trainable parameters: 42,951

Non-trainable parameters: 0

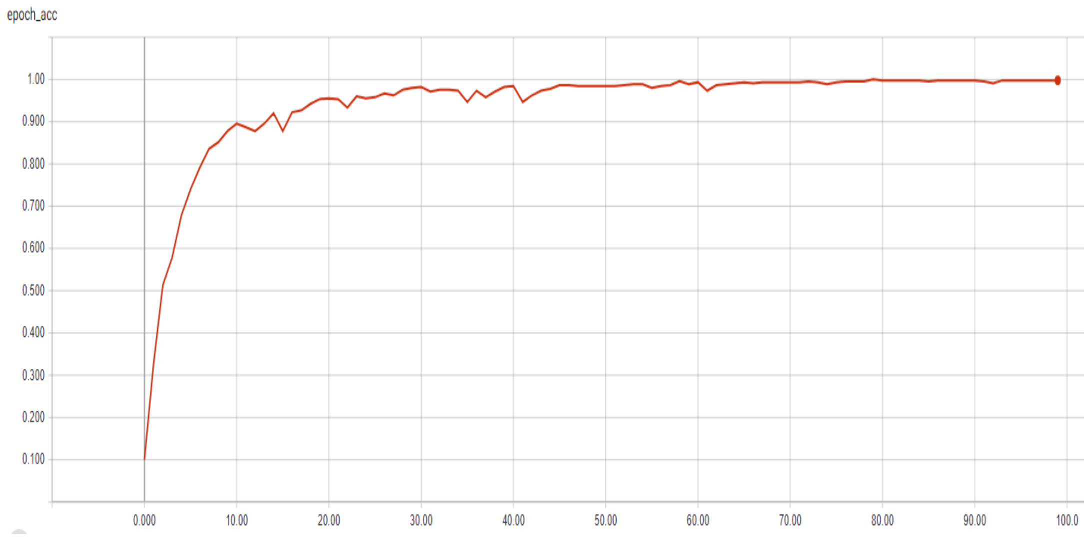


Fig. 4. 1 Graph of training accuracy of 3rd model. Horizontal axis describe number of epoch and vertical axis describe training accuracy. Training accuracy 1 means 100% accuracy 0.4 means 40% accuracy.

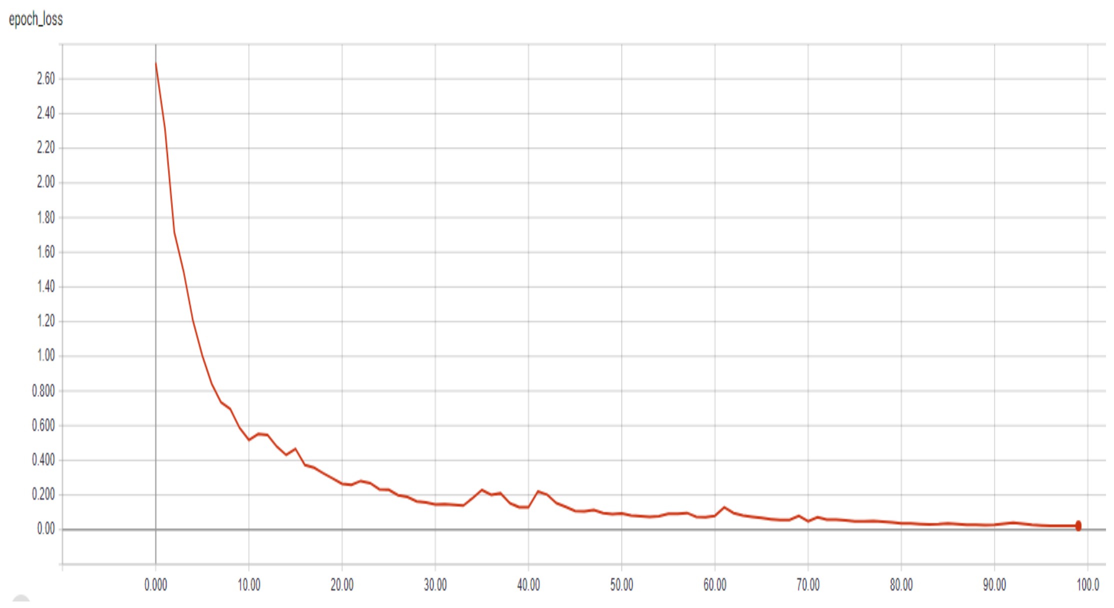


Fig. 4. 2 Graph of training loss of 2nd model. Horizontal axis describe number of epoch and vertical axis describe softmax loss value.

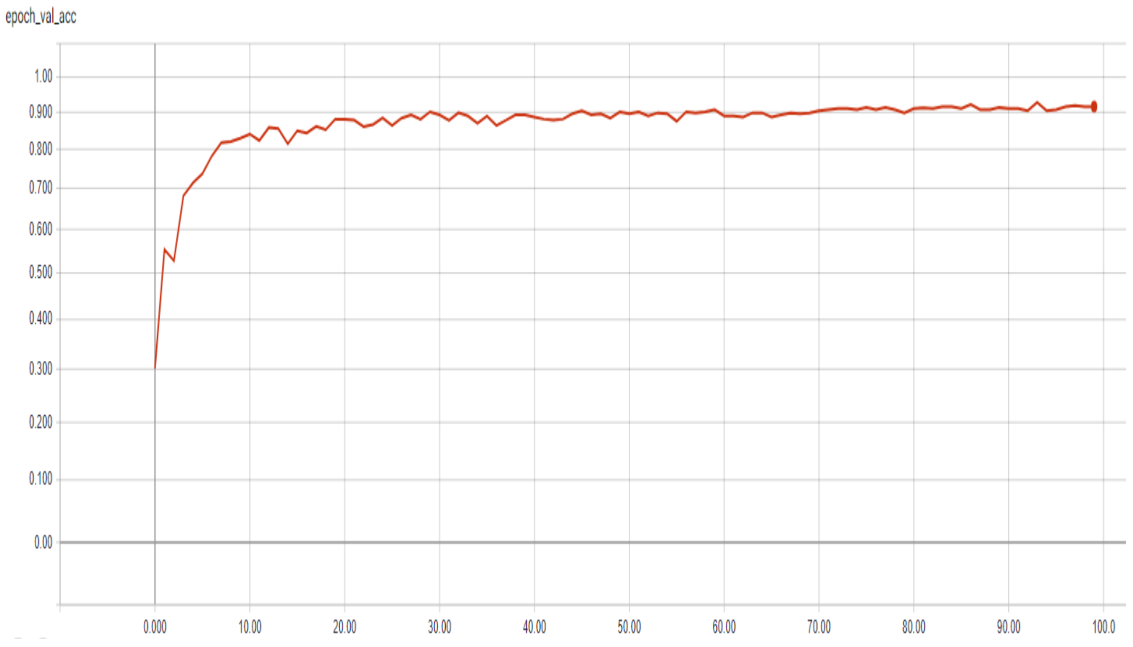


Fig. 4. 3 Graph of validation accuracy of 3rd model. Horizontal axis describe number of epoch and vertical axis describe training accuracy. Training accuracy 1 means 100% accuracy 0.4 means 40% accuracy.

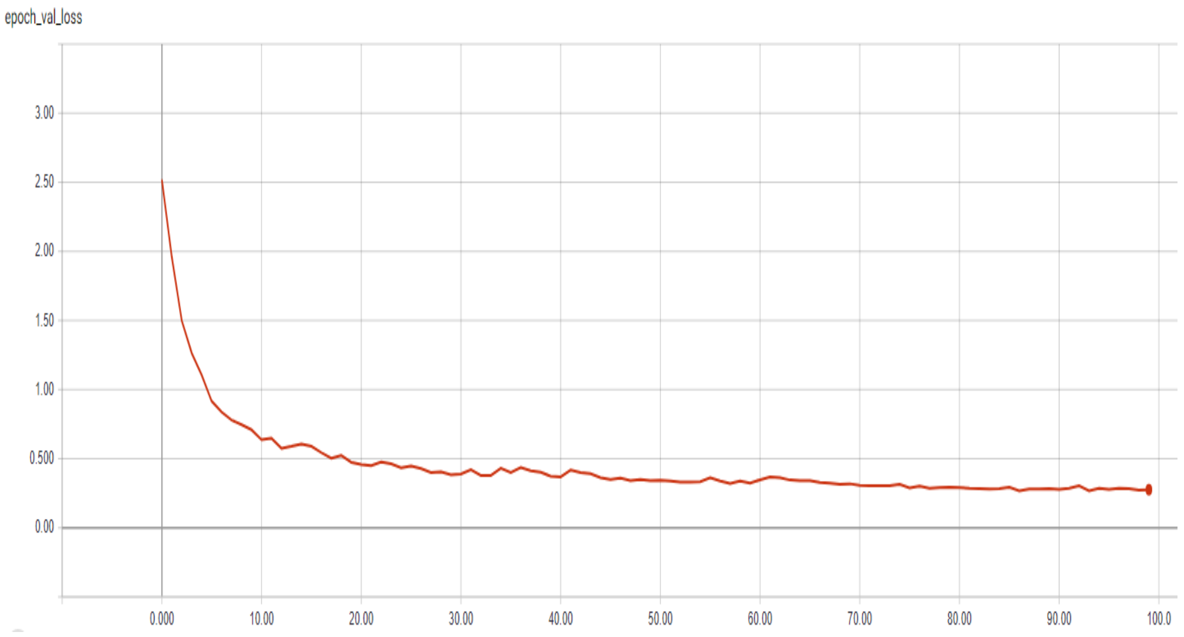


Fig. 4. 4 Graph of validation loss of 3rd model. Horizontal axis describe number of epoch and vertical axis describe softmax loss value.

4.3.4 Architecture of the Fourth Model:

Layer (type)	Output Shape	Parameter #	Activation
input_1	(None, 128, 128, 3)	0	
conv2d	(None, 128, 128, 64)	1792	elu
max_pooling2d	(None, 64, 64, 64)	0	
batch_normalization_v1	(None, 64, 64, 64)	256	
conv2d_1	(None, 64, 64, 32)	18464	elu
max_pooling2d_1	(None, 32, 32, 32)	0	
batch_normalization_v1_1	(None, 32, 32, 32)	128	
conv2d_2	(None, 32, 32, 16)	4624	elu
max_pooling2d_2	(None, 16, 16, 16)	0	

batch_normalization_v1_2	(None, 16, 16, 16)	64	
conv2d_3	(None, 16, 16, 8)	1160	elu
max_pooling2d_3	(None, 8, 8, 8)	0	
batch_normalization_v1_3	(None, 8, 8, 8)	32	
flatten	(None, 512)	0	
dense	(None, 32)	16416	tanh
dense_1	(None, 15)	495	softmax

==

Total parameters: 43,431
Trainable parameters: 43,191
Non-trainable parameters: 240

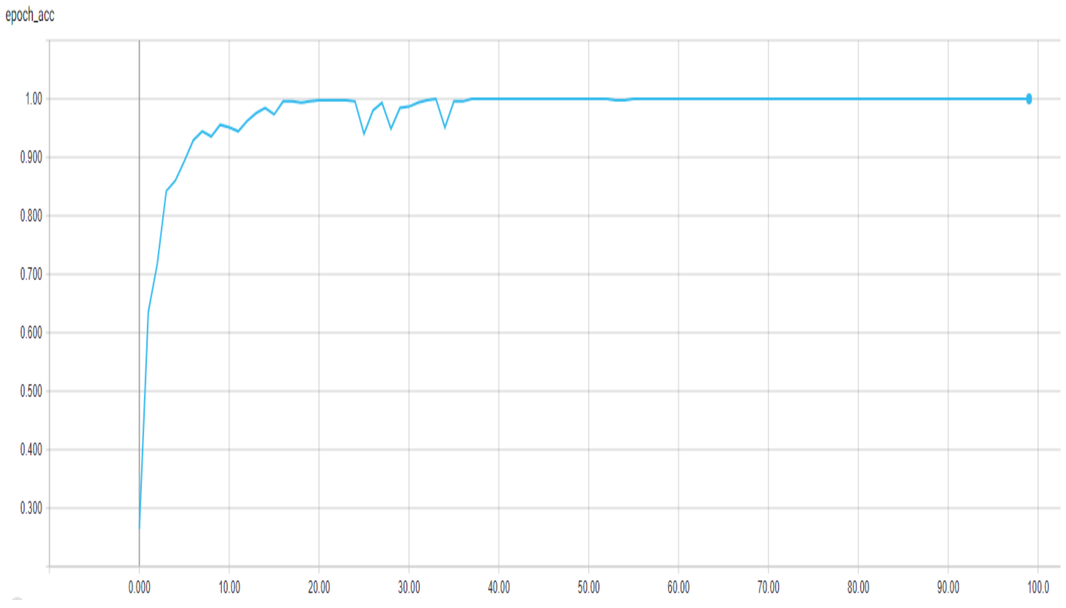


Fig. 4. 5 Graph of training accuracy of 3rd model. Horizontal axis describe number of epoch and vertical axis describe training accuracy. Training accuracy 1 means 100% accuracy 0.4 means 40% accuracy.

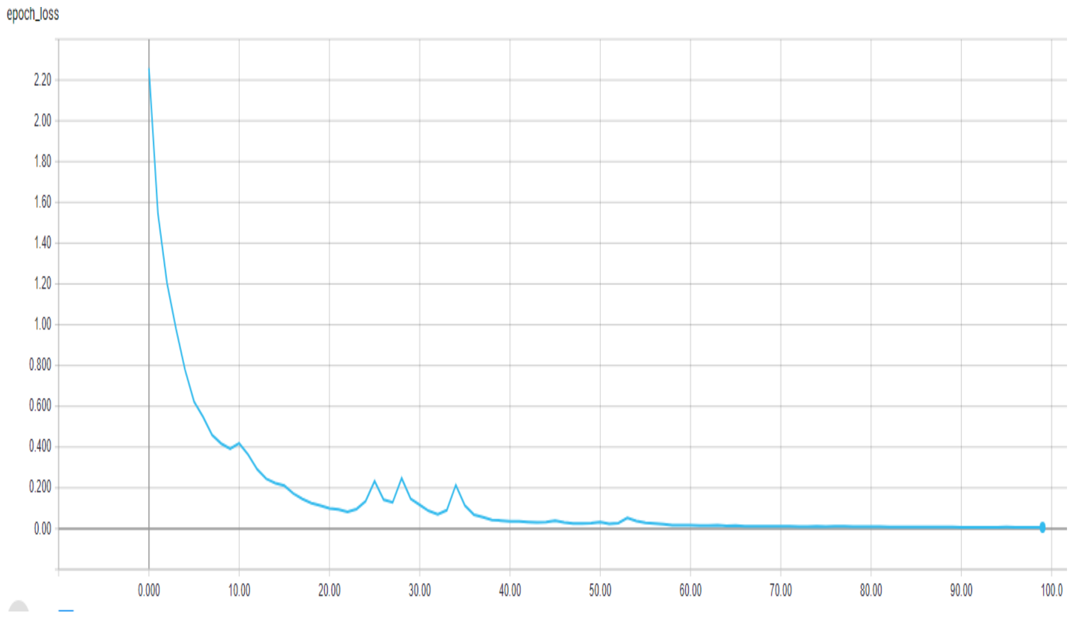


Fig. 4. 6 Graph of training loss of 4th model. Horizontal axis describe number of epoch and vertical axis describe softmax loss value.

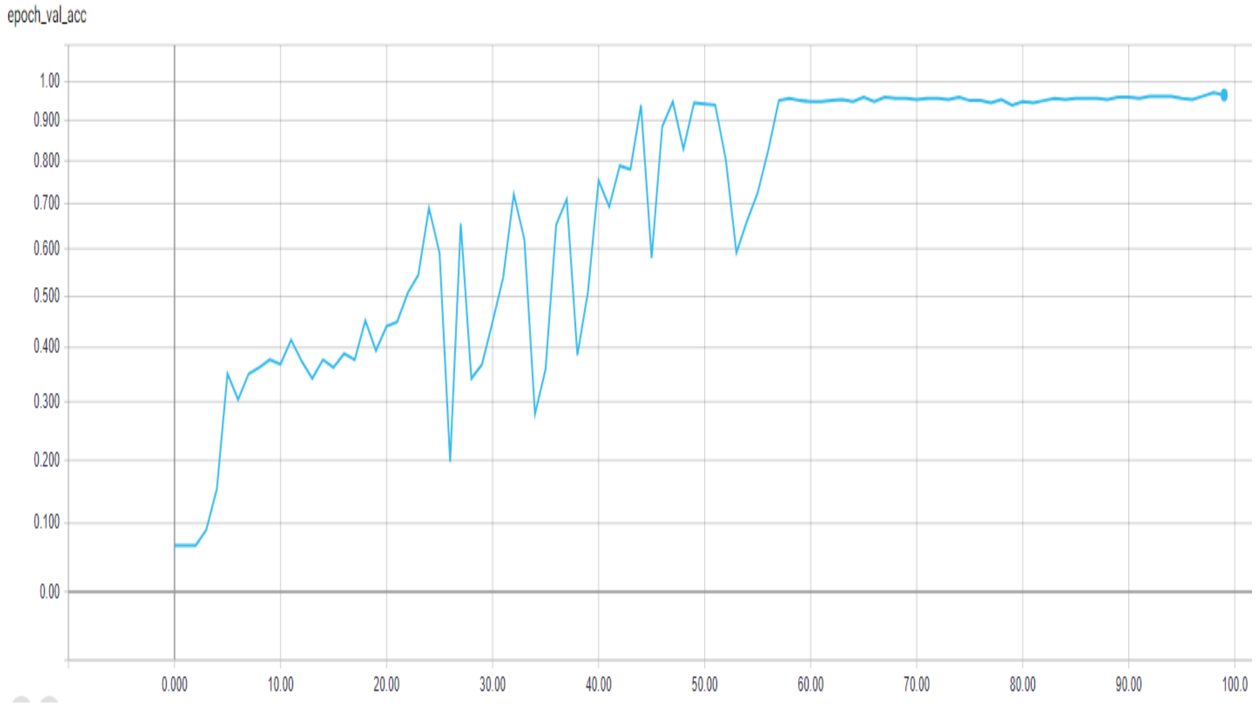


Fig. 4. 7 Graph of validation accuracy of 4th model. Horizontal axis describe number of epoch and vertical axis describe training accuracy. Training accuracy 1 means 100% accuracy 0.4 means 40% accuracy.

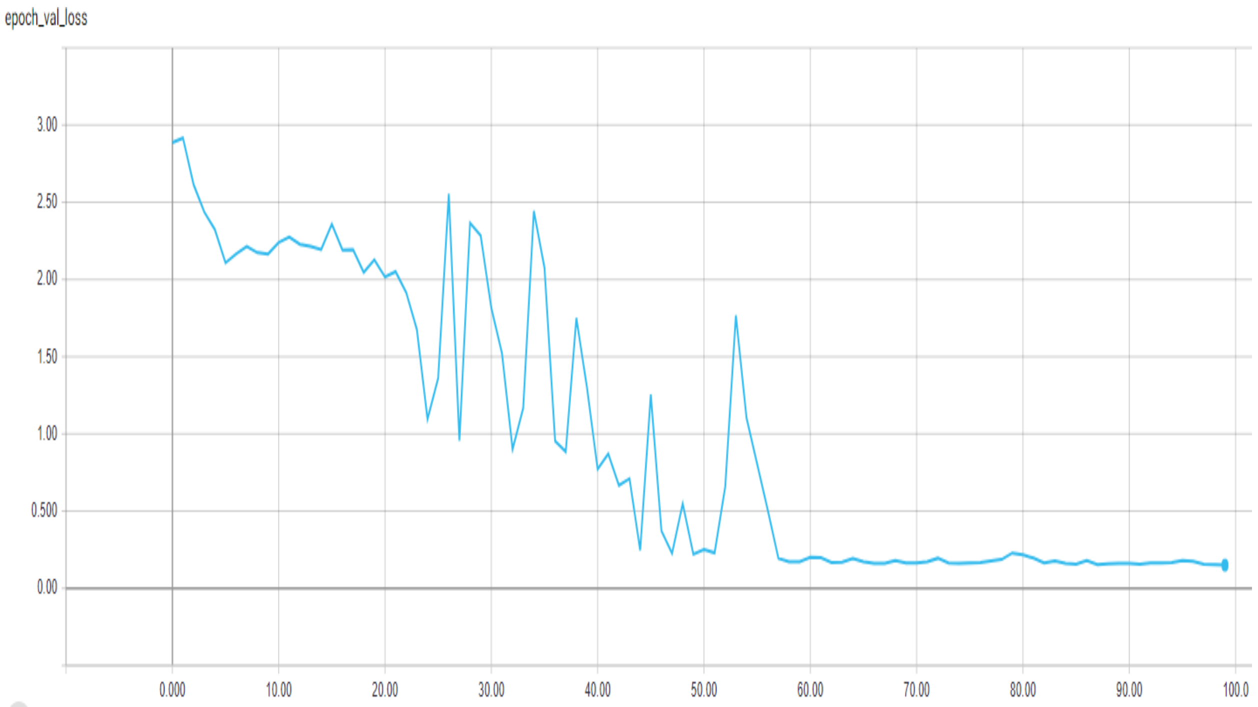
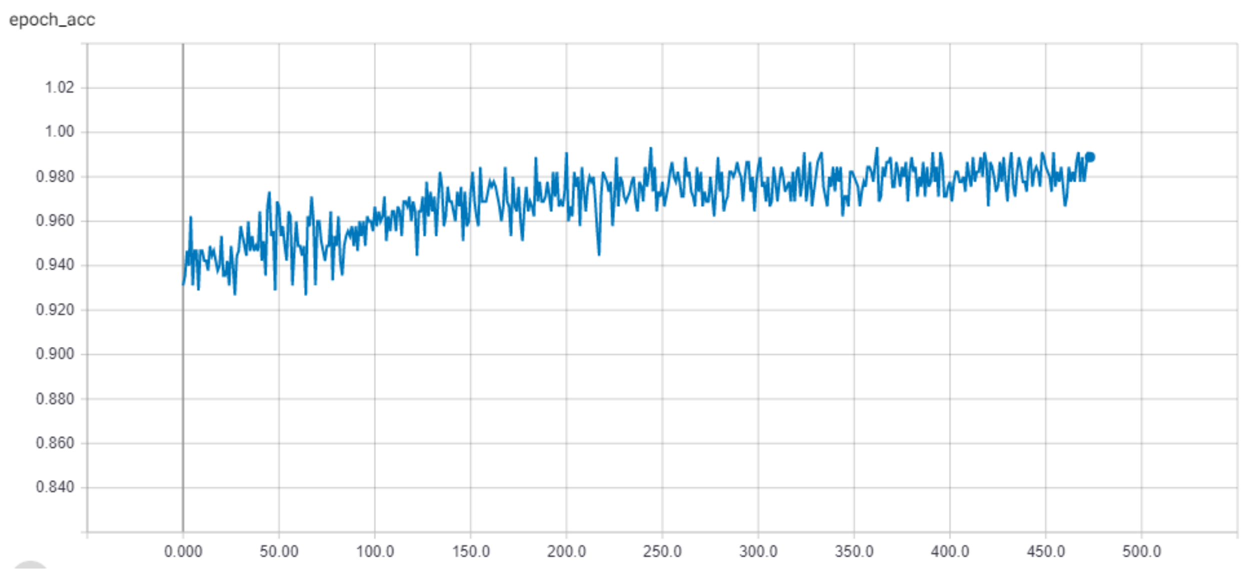


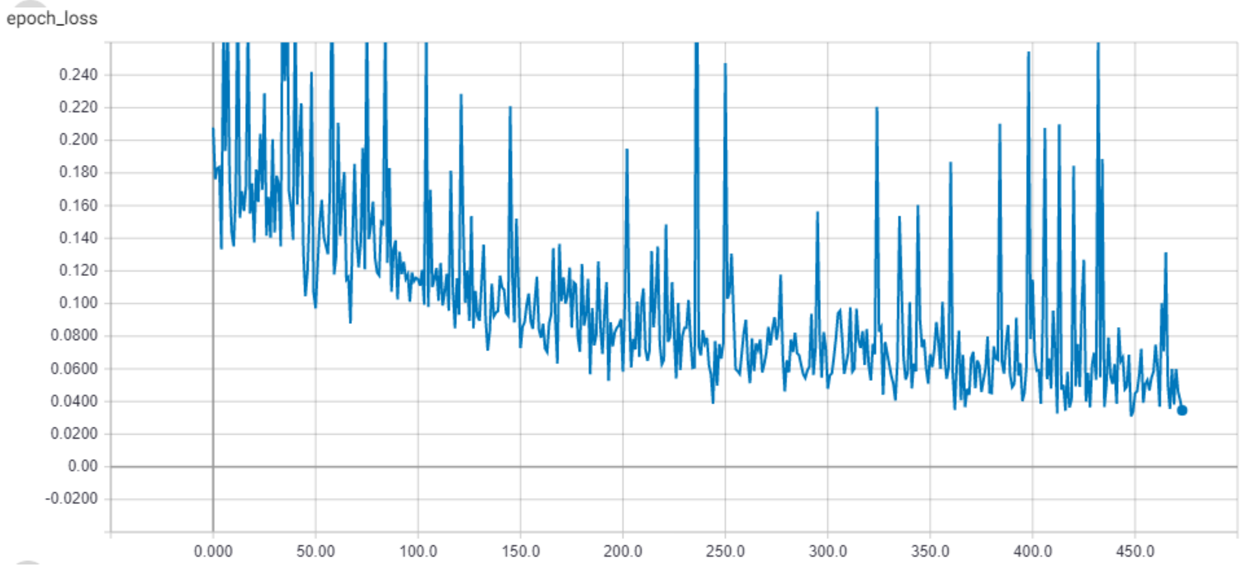
Fig. 4. 8 Graph of validation loss of 4th model. Horizontal axis describe number of epoch and vertical axis describe softmax loss value.

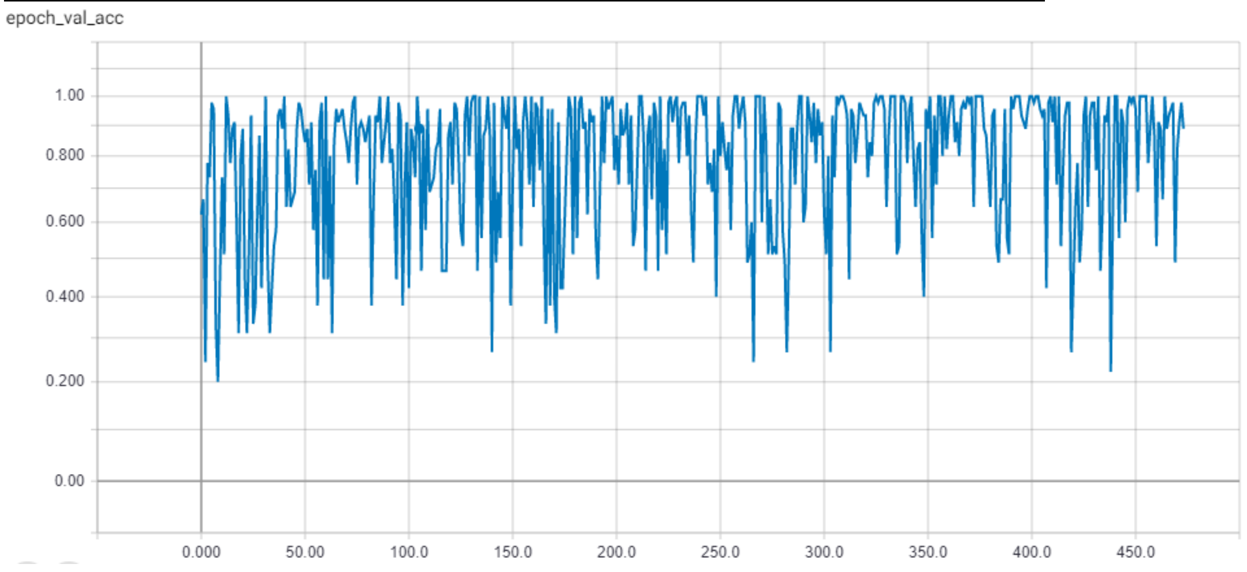
4.3.5 Architecture of the Fifth Model:

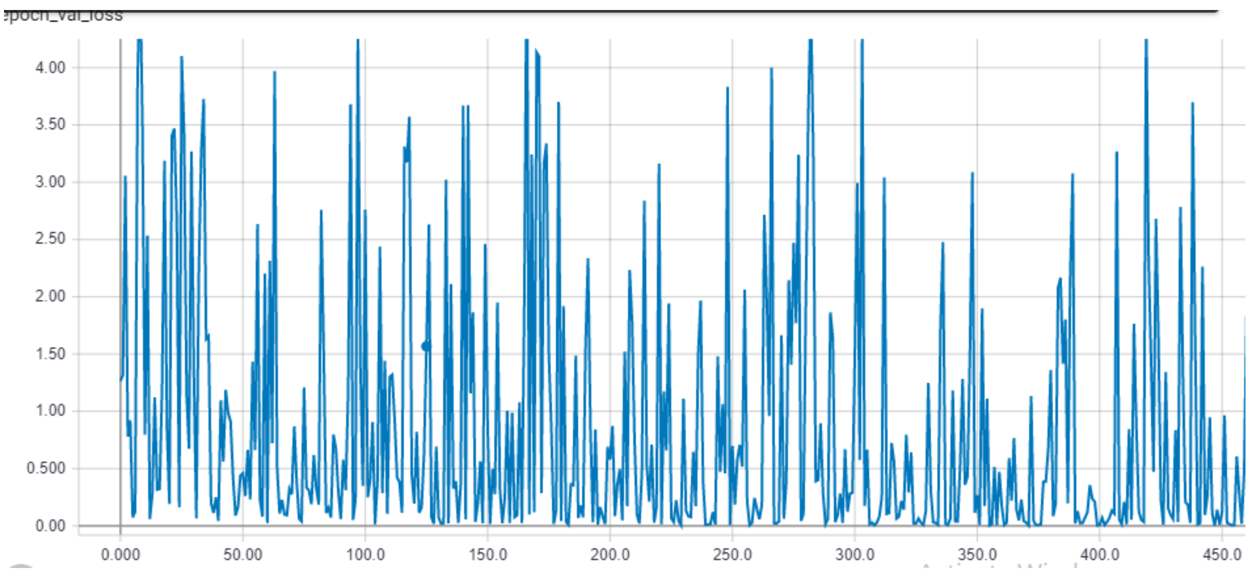
Layer (type)	Output Shape	# of Parameters
input_1	(None, 128, 128, 3)	0
conv2d	(None, 128, 128, 64)	1792
batch_normalization_v1	(None, 128, 128, 64)	256
leaky_relu	(None, 128, 128, 64)	0
max_pooling2d	(None, 64, 64, 64)	0
dropout	(None, 64, 64, 64)	0
conv2d_1	(None, 64, 64, 32)	18464
batch_normalization_v1_1	(None, 64, 64, 32)	128
leaky_re_lu_1	(None, 64, 64, 32)	0

```
max_pooling2d_1            (None, 32, 32, 32)        0
_____________________________________________________________
dropout_1                  (None, 32, 32, 32)        0
_____________________________________________________________
conv2d_2                   (None, 32, 32, 16)        4624
_____________________________________________________________
batch_normalization_v1_2   (None, 32, 32, 16)        64
_____________________________________________________________
leaky_re_lu_2              (None, 32, 32, 16)        0
_____________________________________________________________
max_pooling2d_2            (None, 16, 16, 16)        0
_____________________________________________________________
dropout_2                  (None, 16, 16, 16)        0
_____________________________________________________________
conv2d_3                   (None, 16, 16, 4)         580
_____________________________________________________________
batch_normalization_v1_3 (None, 16, 16, 4)           16
_____________________________________________________________
leaky_re_lu_3              (None, 16, 16, 4)         0
_____________________________________________________________
max_pooling2d_3            (None, 8, 8, 4)           0
_____________________________________________________________
dropout_3                  (None, 8, 8, 4)           0
_____________________________________________________________
flatten                    (None, 256)               0
_____________________________________________________________
dense                      (None, 32)                8224
_____________________________________________________________
dense_1                    (None, 15)                495
=============================================================
Total params: 34,643
Trainable params: 34,411
Non-trainable params: 232
```


epoch_acc
1.02
1.00
0.980
0.960
0.940
0.920
0.900
0.880
0.860
0.840
0.000
50.00
100.0
150.0
200.0
250.0
300.0
350.0
400.0
450.0
500.0


epoch_loss
0.240
0.220
0.200
0.180
0.160
0.140
0.120
0.100
0.0800
0.0600
0.0400
0.0200
0.00
-0.0200
0.000
50.00
100.0
150.0
200.0
250.0
300.0
350.0
400.0
450.0


epoch_val_acc
1.00
0.800
0.600
0.400
0.200
0.00
0.000
50.00
100.0
150.0
200.0
250.0
300.0
350.0
400.0
450.0



4.4 Analysis of Results

From the above results, for the first model as shown Section 4.2.1, from the corresponding results as shown in Figure 4.5, Figure 4.6, Figure 4.7 and Figure 4.8, it has been observed that as in the convolution layer of the model, no activation function is used, training accuracy is lesser and training loss is higher as compared to other 4 models. In first, second, third and fourth model, it is seen that, in spite of the high validation loss, good classification accuracy is obtained using the validation data in. But is has been seen that, the ratio of training loss and validation loss is very high, which shows that the model is overfitting. Also, if we use a model with less number of parameters, then the model does not train well. It is also known to us that, the possible reason for a model to get overfitted is having less number of training data. To overcome this problem, in the proposed fifth model, the dropout layer is added after each activation layer to get a more generalized model. Thus, the problem of overfitting is solved but it causes the oscillation of the validation loss in each epoch. To solve this problem model need to train for more number of epochs for convergence.

Chapter 5

Conclusion

5.1 Limitations of the Work

We tried to train with augmented image by rotateing orginal image in random in random angle. But it shows very training accuracy. Our model unable learn ration invariant feature. Time require train 500 epoch on Intel I7 processor is around 8 hour. We need to stop early before training loss could converge. To prevent network to overfit we need more training data.

5.2 Future Scopes of the Work:

As it has been seen that in the proposed fifth model that the validation loss and validation accuracy is oscillating, so there is a future scope to solve this problem. To increase the size of dataset we can try to generate the dataset by some Generative Neural Model. And for more generalization we can try genetic algorithm based adaptive dropout layer instead of conventional dropout.

References

[1] Lippaman, R.P.(1987). An Introduction to Computing with Neural Nets. IEEE ASSP

[2] Representation, Convolutional Neural Networks and Hidden Markov Models, In Advances in Neural Magazine, Vol.4.,No. 2, pp. 4-22.

[3] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal

[4] Rojas, R. (1996).Neural Networks – A Systematic Introduction, Springer, Berlin (1996)

[5] Press, W.H. et al., Numerical Recipes, Cambridge University Press, Cambridge.

[6] Du J-X, Wang X-F, Zhang G-J (2006) Leaf shape based plant species. Recogn Appl Math Comput 185:883–893. https://doi. org/10.1016/j.amc.2006.07.072

[7] Macleod N, Benfield M, Culverhouse P (2010) Time to automate identification. Nature 467:154–155. https://doi.org/10. 1038/467154a

[8] Babatunde O, Armstrong L, Diepeveen D, Leng J (2015) A survey of computer-based vision systems for automatic identification of plant species. J Agric Inform 6(1):61–71. https://doi. org/10.17700/jai.2015.6.1.152

[9] Cope JS, Corney D, Clark JY, Remagnino P, Wilkin P (2012) Plant species identification using digital morphometrics: a review. Expert Syst Appl 39:7562–7573. https://doi.org/10. 1016/j.eswa.2012.01.073

[10] Ellis B, Ash A, Hickey LJ, Johnson K, Wilf P, Wing S (2009) Manual of leaf architecture. Smithsonian Institution. ISBN: 0-9677554-0-9

[11] Sharma S, Gupta C (2015) A review of plant recognition methods and algorithms. Int J Innov Res Adv Eng (IJIRAE) 2(6):2349-2163

[12] Minu RI, Thyagharajan KK (2011) Automatic image classification using SVM classifier. CIIT Int J Data Min Knowl Eng 3:559–563.

[13] Thyagharajan KK, Minu RI (2013) Prevalent color extraction and indexing. Int J Eng Technol 5(6):4841–4849

[14] [14]9 Thyagharajan KK, Minu RI (2012) Multimodal ontology search for semantic image retrieval. ICTACT J Image Video Process 3:473–478

[15] [15]10 Caglayan A, Guclu O, Can A (2013) A plant recognition approach using shape and color features in leaf images. In: Petrosino A (ed) Image analysis and processing ICIAP 2013, vol-157. Lecture Notes in Computer Science. Springer, Berlin, pp 161–170. https://doi.org/10.1007/978-3-642-41184-7_17

[16] Park J, Hwang E, Nam Y (2008) Utilizing venation features for efficient leaf Image Retrieval. J Syst Softw 81:71–82. https:// doi.org/10.1016/j.jss.2007.05.001

[17] Nam Y, Yung E, Kim D (2008) A similarity based leaf image retrieval scheme and venation feature. J Comput Vis Image Underst 110:245–259. https://doi.org/10.1016/j.cviu.2007.08. 002

[18] Grinblat GL, Uzal LC, Larese MG, Granitto PM (2016) Deep learning for plant identification using vein morphological patterns. Comput Electron Agric 127:418–424. https://doi.org/10. 1016/j.compag.2016.07.003

[19] Bauer J, NikoSunderhauf PP (2007) Comparing several implementations of two recently published feature detectors. Proc Int Conf Intell Autom Syst 40:143–148. https://doi.org/10.3182/ 20070903-3-FR-2921.00027

[20] Lavania S, Matey PS (2014) Leaf recognition using contour based edge detection and sift algorithm. In: 2014 IEEE international conference on computational intelligence and computing research (ICCIC), pp 1–4. https://doi.org/10.1109/iccic. 2014.7238345
[21] Chen Y, Lin P, He Y (2011) Velocity representation method for description of contour based shape classification of weed leaf images. Biosyst Eng 109:186–195. https://doi.org/10.1016/j.bio systemeng.2011.03.004
[22] Laga H, Kurtek S, Srivastava A, Golzarian M, Miklavcic SJ (2012) A Riemannian elastic metric for shape-based plant leaf classification. In: 2012 International conference on digital image computing techniques and applications (DICTA), pp 1–7. https://doi.org/10.1109/dicta.2012.6411702
[23] Mounie S, Yahiaoui I, Verroust Blondet A (2013) A shape based approach for leaf classification using multiscale triangular representation. In: ACM international conference on multimedia retrieval, pp 127–134. https://doi.org/10.1145/12461466. 2461419
[24] Wang B, Brown D, Gao Y, La Salle J (2015) MARCH: a multi scale arch height descriptor for mobile retrieval leaf images. Inf Sci 302:132–148. https://doi.org/10.1016/j.ins.2014.07.028
[25] De Souza MMS, Medeiros FNS, Ramalho GLB, de Paula IC, Oliveria INS (2016) Evolutionary optimization of multiscale descriptor for shape analysis. Expert Syst Appl 63(c):375–385. https://doi.org/10.1016/j.eswa.2016.07.016
[26] Chaki J, Parekh R, Bhattacharya S (2015) Plant leaf recognition using texture and shape features with neural classifiers. Pattern Recogn Lett 58:61–68. https://doi.org/10.1016/j.patrec.2015.02. 010
[27] Cao J, Wang B, Brown D (2016) Similarity based leaf image retrieval using multiscale R-angle description. Inf Sci 374:51–64. https://doi.org/10.1016/j.ins.2016.09.023
[28] Sangle S, Shirsat K, Bhosle V (2013) Shape based plant leaf classification system using android. Int J Eng Res Technol 2(8):1900–1907
[29] Rahmani ME, Amine A, RedaHamou M (2015) Plant leaves classification. In: The first international conference on big data, small data, linked data, open data, pp 75–80. ISBN:978-1- 61208-445-9
[30] Knight D, Painter J, Potter M (2010) Automatic plant leaf classification for a mobile field guide
[31] Thangirala S, Rani J (2015) Perception based on its incline and pier using centroid delineation pitch of leaf. Int J Res Comput Commun Technol 4(3):154–157
[32] Bong MF, Sulong GB, Rahim MSM (2013) recognition of leaf based on its tip and base using centroid contour gradient. IJCSI Int J Comput Sci Issues 10(2):477–482
[33] Fotopoulou F, Laskaris N, Economou G, Fotopoulo S (2013) Advanced leaf image etrieval via multidimensional embedding sequence similarity (mess) method. Pattern Anal Appl 16(3):381–392. https://doi.org/10.1007/s10044-011-0254-6
[34] Belongie S, Malik J, Puzicha J (2002) Shape matching and object recognition using shape contexts. IEEE Trans Pattern Anal Mach Intell 24(4):509–522. https://doi.org/10.1109/34. 993558
[35] Mounie S, Yahiaoui I, Verroust Blondet A (2012) Advanced shape context for plant species Identification using leaf image retrieval. In: ACM international conference on multimedia retrieval. Hongkong, China ACM

[36] Ling H, Jacobs DW (2007) Shape classification using the inner distance. IEEE Trans Pattern Anal Mach Intell 29(2):286–299. https://doi.org/10.1109/tpami.2007.41
[37] Bellhumer PN, Chen D, Feiner S, Jacobs DW, John Kress W, Ling H, Loppez I, Ramamoorthi R (2008) Searching the World's Herbaria: a system for visual identification of plant species. Lecture Notes on Computer Science, pp 116–129
[38] Zhang SW, Zhao MR, Wang XF (2012b) Plant classification based on multilinear independent component analysis. In: Proceedings of the 7th international conference on advanced intelligent computing theories and applications: with aspects of artificial intelligence (ICIC'11), Springer, Berlin, pp 484–490. https://doi.org/10.1007/978-3-642-25944-9
[39] Kumar N, Bellhumer PN, Biswas A, Jacobs DW, Kress WJ, Lopez I, Soares JVB (2012) Leafsnap: a computer vision system for plant species identification. Lecture notes in Computer Science, pp 502–516
[40] Reul C, Toepfer M, Puppe F (2016) Cross dataset evaluation of feature extraction of feature extraction techniques for leaf classification. Int J Artif Intell Appl 7:1–19. https://doi.org/10. 5121/ijaia.2016.7201
[41] Swain KC, Norremark M, Ramus N et al (2011) Weed identification using an automated active shape matching (AASM) technique. Biosyst Eng 110(4):450–457. https://doi.org/10.1016/ j.biosystemseng.2011.09.011
[42] Cerutti G, Tongue L, Coquin D, Vacavant A (2013) Curvature scale based contour understanding for leaf margin shape recognition and species identification. In: International conference on computer vision theory and applications, vol 1, pp 277–284
[43] Du J-X, Huang D-S, Wang X-F, Gu X (2006) Computer-aided plant speciesidentification (CAPSI) based on leafshape matching technique. Trans Inst Meas Control 28(3):275–284
[44] Gwo C-H, Wei YL (2013) Rotary matching of edge features for leaf recognition. Comput Electr Agricu 91:124–134. https://doi. org/10.1016/j.compag.2012.12.005
[45] Prakash N, Sarkar A (2015) Development of shape based leaf categorization. ISOR J Comput Eng 17(1):48–53
[46] Tekkesinoglu S, Rahim MSM, Rehman A, Amin IM, Saba T (2014) Hevea leaves boundary identification based on morphological transformation and edge detection features. Res J Appl Sci Eng Technol 7(12):2447–2451
[47] Corney David PA, Lillian Tang H, Clark JY, Yin H, Jin J (2012) Automating digital leaf measurement: the tooth, the whole tooth, and nothing but the tooth. PLoS ONE 7(8):e42112. https://doi. org/10.1371/journal.pone.0042112
[48] Jin T, Hou X, Li P, Zhou F (2015) A novel method of automatic plant species identification using sparse representation of leaf tooth features. PLoS ONE 10(10):e0139482. https://doi.org/10. 1371/journal.pone.0139482
[49] Cho SI, Lee DS, Jeong JY (2002) Weed plant discrimination by machine vision and artificial network. Bio Syst Eng 83(3):275–280. https://doi.org/10.1006/bioe.2002.0117
[50] Singh K, Gupta I, Gupta S (2010) SVM BDT PNN and Fourier moment technique for classification of leaf shape. Int J Signal Process Image Process Pattern Recogn 3(4):67–78

[51] Dornbusch T, Andrieu B (2010) Lamina2shape—an image processing tool for an eplicit description of lamina shape tested on winter wheat(Triticum aestivum L.). Comput Electron Agric 70:217–224. https://doi.org/10.1016/j.compag.2009.10.009
[52] Hossain J, Amin MA (2010) Leaf shape identification based plant biometrics. In: 2010 13th International conference on computer and information technology (ICCIT), pp 458–463. https://doi.org/10.1109/iccitechn.2010.5723901
[53] Wu SG, Bao FS, Xu EY, Wang Y-X, Cheng Y-F, Xiang Q-L (2007) A leaf recognition algorithm for plant classification using probabilistic neural network. In: IEEE international symposium on signal processing and information technology, pp 1–6. https://doi.org/10.1109/isspit.2007.4458016
[54] Lee KB, Hong KS (2013) An implementation of leaf recognition system using leaf vein and shape. Int J Bio-Sci Bio-Technol 5(2):57–66. https://doi.org/10.1007/978-94-007-5857-5_12
[55] Altartouri H, Abu DA, Maizer A, HashemTamimi RA (2015) Computerized extraction of morphological and geometrical features for plants with compound leaves. J Theor Appl Inf Technol 81(3):474–480
[56] Akif A, Khan MF (2015) Automatic classification of plants based on their leaves. Biosyst Eng 139:66–75. https://doi.org/10. 1016/j.biosystemseng.2015.08.003
[57] Aptoula E, Yanikoglu B (2013) Morphological features for leaf based plant recognition. In: 2013 20th IEEE international conference on image processing (ICIP), pp 1496–1499. https://doi. org/10.1109/icip.2013.6738307
[58] Dutta L, Basu TK (2013) Extraction and optimization of leaves images of mango trees and classification using ANN. Int J Recent Adv Eng Technol 1(3):46–51
[59] Manik FY, Herdiyeni Y, Herliyana EN (2016) Leaf morphlogical feature extraction of digital image Anthocephalus cadamba. TELKOMNIKA 14(2):630–637. https://doi.org/10.12928/tel komnika.v14i2.2675
[60] Chen Y, Lin P, He Y, Zhenghao X (2011) Classification of broadleaf weed images using gabor wavelets and Lie group structure of region covariance on Riemanian manifolds. Biosyst Eng 109:220–227. https://doi.org/10.1016/j.biosystemeng.2011. 04.003
[61] Bruno OM, de Oliveira Plotze R, Falvo M, de Castro M (2008) Fractal dimension applied to plant identification. Inf Sci 178(12):2722–2733. https://doi.org/10.1016/j.ins.2008.01.023
[62] De Oliveira R, Plotze MF, Pa´dua JG, Bernacci LC, Vieira MLC, Oliveira GCX, Bruno OM (2005) Leaf shape analysis using the multiscale minkowski fractal dimension, a new morphometric method : a study with Passiflora. Can J Bot 83:287–301. https:// doi.org/10.1139/B05-002
[63] Muchtar M, Suciati N, Fatichah C (2016) Fractal dimension and lacunarity combination for plant leaf classification. J Comput Sci Inf 9(2):96–105. https://doi.org/10.21609/jiki.v912.385
[64] [86]59 Vijayalakshmi B, Mohan V (2016) Kernel based PSO and FRVM: an automatic plant leaf type detection using texture, shape and color features. Comput Electron Agric 125:9–112. https://doi.org/10.1016/j.compag.2016.04.033
[65] Venkatesh SK, Raghavendra R (2011) Local gabor phase quantization scheme for robust leaf classification. In: 2011 Third national conference on computer vision, pattern recognition, image processing and graphics (NCVPRIPG), pp 211–214. https://doi.org/10.1109/ncvpripg.2011.52

[66] Qi X, Xiao R, Li C-G, Qiao Y, Guo J, Tang X (2014) Pairwise rotation invariant co-occurrence local binary pattern. IEEE Tran Pattern Anal Mach Intell 36:2199–2212. https://doi.org/10.1109/ tpami.2014.2316826

[67] Qiuyan L,Wenfa Q (2015) Multiscale local binary pattern based on path integral for texture classification. In: IEEE international conference on image processing, pp 26–30. https://doi.org/10. 1109/icip.2015.7350752

[68] Sumathi CS, Senthil Kumar AV (2012) Edge and texture fusion for plant leaf classification. Int J Comput Sci Telecommun 3(6):6–9

[69] Carranza Rojas J, Mata Montero E (2016) Combining leaf shape and texture of costa rica plant species identification. CLEI Electr J 19(7):1–29. https://doi.org/10.19153/cleiej.19.1.7

[70] Siricharoen P, Scotney B, Morrow P, Parr G (2016) A lightweight mobile system for crop disease diagnosis. In: International conference image analysis and recognition, pp 783–791

[71] [103]67 Zhang J, Zhao H, Liang J (2013) Continuous rotation invariant local descriptors for texton dictionary-based texture classification. Comput Vis Image Underst 117(1):56–75

[72] Minu RI, Thyagharajan KK (2014) Semantic rule based image visual feature ontology creation. Int J Autom Comput 11(5):489–499. https://doi.org/10.1007/s11633-014-0832-3

[73] Minu RI, Thyagharajan KK (2012) A novel approach to build image ontology using texton. Advances in Intelligent Systems and Computing, vol 182, pp 333–339. Springer, Berlin. ISBN: 978-3-642-32062-0 (Print) 978-3-642-32063-7 (Online), ISSN: 2194-5357

[74] Guo Z, Li Q, Zhang L, You J, Zhang D, Liu W (2013) Is local dominant orientation necessary for the classification of rotation invariant texture? Neuro Computing 116:182–191. https://doi. org/10.1016/j.neucom.2011.11.038

[75] Trczinksi T, Christoudias M, Fua P, Lepetit V (2013) Boosting binary key point descriptors. IEEE Conf Comput Vis Pattern Recogn. https://doi.org/10.1109/CVPR2013.370

[76] Le TL, Tran D-T, Hoang V-N (2014) Fully automatic leaf based plant identification, application of Vietnamese medicinal plant search. In: Proceedings of the fifth symposium on information and communication technology, pp 146–154. https://doi.org/10. 1145/2676585.2676592

[77] Le TL, Tran D-T, Hoang V-N (2014) Kernel descriptor based plant leaf identification. Image Process Theory Tools Appl. https://doi.org/10.1109/ipta.2014.7001990

[78] Adsule BR, Bhattad JM (2015) Leaves classification using SVM using neural network disease identification. Int J Innov Res Comput Commun Eng 3(6):5488–5495

[79] Sainin MS, Alfred R (2009) Half leaf shape feature extraction for leaf identification. In: First Malaysian international conference on artificial intelligence

[80] Bagalkote IS, Vibhute AS, More BM (2014) Texture analysis using DWT for grape plant species classification. J Bot Sci 3(3):34–40

[81] Anami BS, Pujari JD, Yakkundimath R (2011) Identification and classification of normal and affected agriculture/horticulture produce based on combined color and texture feature extraction. Int J Comput Appl Eng Sci 1(3):356–360

[82] Sathish V, Ramesh K (2015) Identification and classification of plant leaf disease. Int J Adv Res Sci Eng 4(1):978–983

[83] Ravisankar AM, Mohanapriya M (2016) Classification of name based on leaf recognition using BT and ED algorithm. Int J Comput Appl Technol Res 5(4):191–197

[84] Nandyal S, Bagewadi S (2013) Automated identification of plant species from images of leaves and flowers used in the diagnosis of arthritis. Int J Res Eng Adv Technol 1(5):1–10

[85] Zhai C-M, Du J-X (2008) Applying extreme learning machine to plant species identification. In: International conference on information and automation, 2008. ICIA 2008, pp 879–884. https://doi.org/10.1109/icinfa.2008.4608123

[86] Du J-X, Zhai C-M, Wang Q-P (2013) Recognition of plant leaf Image based on fractal dimension features. Neuro Comput 116:150–156. https://doi.org/10.1016/j.neucom.2012.03.028

[87] Rahmani ME, Amine A, RedaHamou M (2015) Plant leaves classification. In: The first international conference on big data, small data, linked data, open data, 75–80. ISBN:978-1-61208- 445-9

[88] Arunpriya C, Thanamani AS (2015) Fuzzy inference system algorithm of plant classification for tea leaf recognition. Indian J Sci Technol 8(S7):179–184

[89] Wang X-F, Huang D-S, Ji-Xiang D, Huan X, Heutte L (2008) Classification of plant leaf images with complicated background. Appl Math Comput 205:916–926

[90] Kadir A (2015) Leaf identification using fourier descriptors and other shape features. Gate Comput Vis Pattern Recogn 1(1):3–7. https://doi.org/10.15579/gtcvpr.0101.003007.

[91] Sule M, Matas J (2014) Texture based leaf identification. Research Report of CMP, Crez Technical University. (10):CTUCMP-2014-10

[92] Narayan V, Subbarayan G (2014) An optimal feature subset selection using GA for leaf classification. Int Arab J Inf Technol 11(5):447–451.

[93] Golzarian MR, Frick RA (2011) Classification of images of wheat, ryegrass and brome grass species at early growth stages using principal component analysis. Plant Methods 7:28

[94] Abdolvahab Eshani Rad (2010) Plant classification based on leaf recognition. Int J Comput Sci Inf Secur 8(4):78–81

[95] Valliammal N, Geethalakshmi SN (2012) An optimal feature subset selection for leaf analysis. World Acad Sci Eng Technol 62(2012):440–445

[96] Sainin MS, Ahmad F, Alfred R (2016) Improving the identification and classification of Malaysian medicinal leaf images using ensemble method. In: International conference on ICT for transformation, pp 1–6

[97] Sainin MS, Alfred R, Ghazali TK (2014) Malaysian medicinal plant leaf shape identification and classification. In: Knowledge management international conference, pp 578–583

[98] Dyrmann M, Karstof H, Midity HS (2016) Plant species classification using deep convolutional network. Biosyst Eng 151:72–80. https://doi.org/10.1016/j.biosystemeng.2016.08.24

[99] Sladojevic S, Arsenovic M, Andala A, Glibrt D, Stefanvoic D (2016) Deep neural networks based recognition of plant diseases by leaf image classification. Comput Intell Neurosci. https://doi. org/10.1115/2016/3289801