PERFORMANCE ANALYSIS OF PLANT SPECIES RECOGNITION USING CONVENTIONAL AND DEEP LEARNING TECHNIQUES

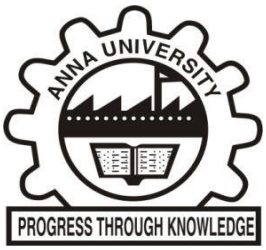
A THESIS

Submitted by

ANUBHA PEARLINE S

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY



FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

ANNA UNIVERSITY

CHENNAI 600 025

MAY 2021

ANNA UNIVERSITY
CHENNAI 600 025

BONAFIDE CERTIFICATE

The research work embodied in the present Thesis entitled "**PERFORMANCE ANALYSIS OF PLANT SPECIES RECOGNITION USING CONVENTIONAL AND DEEP LEARNING TECHNIQUES**" has been carried out in the Department of Electronics Engineering, Madras Institute of Technology Campus, Anna University, Chennai. The work reported herein is original and does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion or to any other scholar.

I understand the University's policy on plagiarism and declare that the thesis and publications are my own work, except where specifically acknowledged and has not been copied from other sources or been previously submitted for award or assessment.

ANUBHA PEARLINE S
RESEARCH SCHOLAR

DR. V. SATHIESH KUMAR
SUPERVISOR
Assistant Professor
Department of Electronics Engineering
Madras Institute of Technology Campus
Anna University
Chennai - 600 044.

ABSTRACT

Real-time Plant Species Recognition under unconstrained environments is challenging and time-consuming process. Two factors significantly affect the efficacy of the recognition system (using leaf). They include the complex nature of leaves and the challenges associated with computer vision methods. Complex nature is due to the vast plant diversity, intra-class variations, leaf structure, leaf color (seasonal variation, aging factor), leaf type (simple and compound), geometry, and venation properties. The challenges associated with computer vision methods primarily relate to the image acquisition process. The captured images exhibit illumination variations, scale changes, orientation or viewpoint modifications, and different backgrounds. Furthermore, in most of the reported literature, plant species recognition is performed using the datasets of foreign origin. These datasets do not include the plant images of Indian origin.

This research work proposes an efficient system for plant species recognition in real-time. A custom-developed dataset named ‘Leaf-12’ is formed using the Indian plant species. The dataset includes twelve plant species. The images in the dataset are captured by varying the illumination, scale changes, orientation or viewpoint modification, and different backgrounds. Four approaches are tested for real-time plant species recognition. They include the Conventional image processing method, PReLU based Backpropagation Neural Network (P-BPNN), Single Deep Learning (Convolutional Neural Network (CNN)) Architectures, and Dual Deep Learning Architectures (DDLA). The methods are evaluated using four datasets, namely, Flavia, Folio, Swedish leaf, and custom-developed Leaf-12.

In the conventional image processing method, the feature extraction is carried out by obtaining the color, shape, and texture features. Color Channel Statistics (CCS) method is used to extract the leaf color information. The shape features are obtained by Hu Moments (HM). The texture information is acquired

from Local Binary Pattern (LBP) and Haralick Texture (HT) methods. The extracted features are then concatenated to form the final feature vector. The final feature vector is termed as CSTF (Color, Shape, and Texture Features). Then, the CSTF feature vector is classified using several machine learning classifiers, namely, Linear Discriminant Analysis (LDA), multinomial Logistic Regression (LR), Naïve Bayes (NB), K-Nearest Neighbor (K-NN), Classification and Regression Tree (CART), Random Forest Classifier (RFC), Bagging Classifier (BC) and Support Vector Machine (SVM). From the experimental investigation, it is observed that the CSTF method with the Random Forest classifier achieved an accuracy of about 82.38% (Leaf-12 dataset).

In the second approach, a custom-created P-BPNN (inclusive of two hidden layers) is employed for plant species recognition. PReLU activation function is used in hidden layers. Also, the optimization of hyperparameters (Optimizer, Learning Rate, Activation Function, Alpha in PReLU activation function, and Number of epochs) is carried out in order to improve the performance of the plant species recognition system. From the experimental analysis, it is observed that the Shallow Neural Network (H1 Neurons = 500, H2 Neurons = 250) with optimized hyperparameters (Adam Optimizer, Learning Rate = 0.001, PReLU Activation function with α value of 0.1, Number of epochs = 50) resulted in an accuracy of 87.50% (Leaf-12 dataset).

In the third method, two approaches for single pre-trained Deep Learning (DL) architectures are tested for plant species recognition. The two approaches include CNN as feature extractor + Machine Learning classifiers (Approach-1) and fine-tuned CNN architectures (Approach-2). Experiments are conducted for various CNN architectures such as VGG-16, VGG-19, Inception-V3, ResNet-50, Inception ResNet-V2, Xception, MobileNet, DenseNet-121, DenseNet-169 and DenseNet 201. The Machine Learning classifiers used for Approach-1 are Linear Discriminant Analysis (LDA), multinomial Logistic Regression (LR), Naïve Bayes (NB), K-Nearest Neighbor (K-NN), Classification and Regression Tree (CART), Random Forest Classifier (RFC), Bagging Classifier (BC), Multi-Layer Perceptron (MLP) and Support Vector Machine

(SVM). From the experimental investigation, it is observed that the fine-tuned DenseNet-169 CNN architecture resulted in an accuracy of 99% (Leaf-12 dataset). But, it resulted in higher computational time. It also resulted in a large number of mispredictions, when the methodology is exposed to real-time leaf images.

In the fourth method, two CNN architectures are connected in parallel to form a Dual Deep Learning Architecture (DDLA). Three CNN architectures are chosen based on their properties to form a DDLA. The three CNN architectures include MobileNet (Lightweight architecture), Xception (Separable convolutions with Residual connections), and DenseNet-121 (Feature reuse property). The features are extracted using DDLA and then classified using Machine Learning classifiers. The Machine Learning classifiers used for the analysis are Linear Discriminant Analysis (LDA), multinomial Logistic Regression (LR), Naïve Bayes (NB), K-Nearest Neighbor (K-NN), Classification and Regression Tree (CART), Random Forest Classifier (RFC), Bagging Classifier (BC), Multi-Layer Perceptron (MLP) and Support Vector Machine (SVM).

Two approaches are employed to perform the DDLA analysis, namely, similar DLs and dissimilar DLs. Similar DL architectures utilize the same CNN architecture in both the branches of DDLA. The similar DLs used are MobileNet + MobileNet CNN (S1-DDLA), Xception + Xception CNN (S2-DDLA) and DenseNet-121 + DenseNet-121 CNN (S3-DDLA). Dissimilar DLs utilize two different CNN architectures in the branches of DDLA. The dissimilar DLs used are MobileNet + Xception CNN (DDLA-1), Xception + DenseNet-121 CNN (DDLA-2) and MobileNet + DenseNet-121 CNN (DDLA-3). From the experimental analysis, it is observed that the DDLA-3 (MobileNet + DenseNet-121) with multinomial Logistic Regression (LR) classifier resulted in an accuracy of 99.39% (Leaf-12 dataset). The usage of DDLA with LR classifier significantly reduces the number of mispredictions on real-time leaf images. Hence, this method is highly suitable for real-time plant species recognition under unconstrained environments.

ACKNOWLEDGMENT

I owe my deepest gratitude and profound thanks to my research supervisor **Dr. V. Sathiesh Kumar**, Assistant Professor, Department of Electronics Engineering, Madras Institute of Technology Campus, Anna University, Chennai, for his inspiring guidance, patience, constant motivation, and enthusiasm. Without his persistent help, this thesis would not be possible.

I would like to thank my Doctoral Committee members, **Dr. K. Boopathy Bagan**, Retired Professor, Department of Electronics Engineering, Madras Institute of Technology Campus, Anna University, and **Dr. B. Vijayalakshmi**, Professor, B. S. Abdur Rahman Crescent Institute of Science and Technology for their insightful suggestions and support during my research. I also thank **Dr. M. Ganesh Madhan**, Head of the Department, and other staff from the Department of Electronics Engineering, Madras Institute of Technology Campus, Anna University.

I express my gratitude to my research colleagues and Post Graduate students from Opto-Mechatronics laboratory, Department of Electronics Engineering, Madras Institute of Technology Campus, Anna University, for their encouragement and support during this research. Especially, I thank Ms. S. Harini, M.E. VLSI (2016-2018) for helping me in data collection.

I would like to thank God Almighty for giving me the strength, knowledge, ability, and opportunity to undertake this research and complete it satisfactorily. Finally, my deep and sincere gratitude to my family for their continuous and incessant love, help, and support. I thank my sister for always being with me as a friend. I am forever indebted to my parents for giving me the opportunities and experiences that have made me who I am.

ANUBHA PEARLINE S

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

APIS	-	Advanced Plant Identification System
ASM	-	Angular Second Moment
ACO	-	Ant Colony Optimization
ALID	-	Area of Least Inertia Distance
AUC	-	Area Under Curve
ANN	-	Artificial Neural Network
AE	-	AutoEncoder
AP	-	Average Pooling
BPNN	-	Backpropagation Neural Network
BoW	-	Bag of Words
BC	-	Bagging Classifier
B/T	-	Base and Tip model
BN	-	Batch Normalization
BS	-	Batch Size
b	-	bias
$\hat{m}_t$	-	Bias-corrected First Moment
$\hat{v}_t$	-	Bias-corrected Second Moment
C	-	C^{th} Class
ψ_{ij}	-	Central Moment
CD	-	Centroid Distance
C-R	-	Centroid-Radii
CART	-	Classification and Regression Tree
Cl	-	Classifier
CCS	-	Color Channel Statistics
CAPSI	-	Computer-Aided Plant Species Identification

CSTF	-	Color, Shape and Texture Features
CAT	-	Concatenation
P(A/B)	-	Conditional Probability
CBR	-	Convolution-Batch Normalization-ReLU
Conv(i, j)	-	Convolution Matrix
CNN	-	Convolutional Neural Network
C-RNN	-	Convolutional-Recurrent Neural Network
CUR-WKSR	-	CUR decomposition-Weighted Kernel Sparse Representation
HCS	-	Curvature Scale Histogram
DCNN	-	Deep Convolutional Neural Network
DN	-	Deconvolution Network
DL	-	Deep Learning
DWC	-	DepthWise Convolution
DWSC	-	DepthWise Separable Convolution
DFT	-	Discrete Fourier Transform
DDLA	-	Dual Deep Learning Architecture
DS-LBP	-	Dual Scale Local Binary Pattern
H(E)	-	entropy
EnS	-	Entropy Sequence
ENVIS	-	Environmental Information System
ep	-	epochs
E	-	Error
EM	-	Expectation Maximization
ELU	-	Exponential Linear Unit
F1	-	F1-score
FN	-	False Negative

FP	-	False Positive
FFT	-	Fast Fourier Transform
X	-	Feature Vector
FFNN	-	Feed Forward Neural Network
FV	-	Fisher Vector
FCA	-	Ficus deltoidea, Citrus, Acanthaceae
FCL	-	Fully Connected Layer
GRNN	-	General Regression Neural Network
GAN	-	Generative Adversarial Networks
GAP	-	Global Average Pooling
g_t	-	gradient
GLCM	-	Gray Level Co-occurrence Matrix
l	-	Ground truth label
GCNN	-	Growing Convolutional Neural Network
ρ	-	Growth Rate
GPU	-	Graphics Processing Unit
HT	-	Haralick Texture
HOG	-	Histogram of Gradients
HCoS	-	Histogram of Curvature over Scale
HM	-	Hu Moments
HSV	-	Hue, Saturation, and Value
HGO-CNN	-	Hybrid Generic Convolutional Neural Network
k	-	Hyperparameter in K-NN
f(m, n)	-	Image
IB	-	ImageNet Biases
ILVSRC	-	ImageNet Large Scale Visual Recognition Challenge
IW	-	ImageNet Weights

I-IDSC	-	Independent-Inner Distance Shape Context
$T(.)$	-	Indicator function
u_t	-	Infinity Norm Exponentially Weighted
x_f	-	Input Features
ICL	-	Intelligent Computing Laboratory
K-NN	-	K-Nearest Neighbor
K	-	Kernel or Filter or Weight Matrix
KDES	-	Kernel Descriptor
KL	-	Kullback-Leibler
L	-	Layer
LReLU	-	Leaky Rectified Linear Unit
η	-	Learning rate
LDA	-	Linear Discriminant Analysis
LAII	-	Local Area Integral Invariant
LBP	-	Local Binary Pattern
LBPV	-	Local Binary Pattern Variance
LMC	-	Local Mean based Clustering
LSCL	-	Local Similarity-based Classification Learning
LWRSC	-	Local Sparse Representation based Classification
LLDE	-	Locally Linear Discriminant Embedding
LLE	-	Locally Linear Embedding
LR	-	Logistic Regression
ML	-	Machine Learning
MAX	-	Maximum
MGLLDP	-	Maximum Gap Local Line Direction Patterns
MP	-	Max Pooling
M-P	-	McCulloch and Pitt’s neuron

μ	-	Mean
MIN	-	Minimum
M	-	Model
MDP	-	Modified Dynamic Programming
MLLDE	-	Modified Locally Linear Discriminant Embedding
MMMC	-	Modified Maximizing Margin Criterion
M-I	-	Moment Invariants
MMC	-	Move Median Center
MCNNA	-	Multi-branch Convolutional Neural Network with Attention
MLP	-	Multi-Layer Perceptron
MARCH	-	Multiscale Arch Height
MDM	-	Multi-scale Distance Matrix
MSF-CNN	-	Multi-Scale Fusion Convolutional Neural Network
MNHN	-	Muséum National d'Histoire Naturelle
NB	-	Naïve Bayes
ne_i	-	neighboring image
z_{in}	-	Net Input
NFC	-	Neuro-Fuzzy Controller
NC	-	Normal Convolution
ϕ_{ij}	-	Normalized Central Moment
$p(i, j)$	-	Normalized Gray Matrix
nc	-	normalizing constant
c	-	Number of classes
Gr	-	Number of gray levels
N	-	Number of images
pt	-	Number of points

O	-	Output Dimensions
pd	-	padding
PReLU	-	Parametric Rectified Linear Unit
PDA	-	Penalized Discriminant Analysis
PSR	-	Plant Species Recognition
PWC	-	PointWise Convolution
PFT	-	Polar Fourier Transform
Pr	-	Precision
y_i	-	Predicted Label
P-BPNN	-	PReLU based Backpropagation Neural Network
α	-	PReLU hyperparameter
PCA	-	Principal Component Analysis
PNN	-	Probabilistic Neural Network
$P(A)$	-	Probability
PSLM	-	Progressive Sample Learning Method
PCNN	-	Pulse-Coupled Neural Network
q_i	-	query image
RBF	-	Radial Basis Function
RBPNN	-	Radial Basis Probabilistic Neural Network
r	-	radius
RFC	-	Random Forest Classifier
Re	-	Recall
ReLU	-	Rectified Linear Unit
RGB	-	Red, Green, and Blue
ResNet	-	Residual Networks
RL	-	Run Length
RNN	-	Recurrent Neural Network

ROI	-	Region of Interest
ROM-LBP	-	Region Overall Mean-Local Binary Pattern
SIFT	-	Scale-Invariant Feature Transform
SF	-	Shape Features
S-Inception	-	Siamese-Inception
SVD	-	Singular Value Decomposition
F	-	Size of the filter
SR	-	Sparse Representation
SSO	-	Spatial Structure Optimizer
σ	-	Standard deviation
SGD	-	Stochastic Gradient Descent
s	-	stride
SLLE	-	Supervised LLE
SVM	-	Support Vector Machine
TF	-	Texture Features
Top-1	-	Top-1 accuracy
Top-5	-	Top-5 accuracy
TCM	-	Traditional Chinese Medicine
TN	-	True Negative
TP	-	True Positive
TA-CNN	-	Two-way Attention-based Convolutional Neural Network
UHMT	-	Unconstrained Hit and Miss Transform
UoM	-	University of Mysore
VGG	-	Very large Convolutional Network
VLAD	-	Vector of Locally Aggregated Descriptors
w_i	-	weight vector

CHAPTER 1

INTRODUCTION TO PLANT SPECIES RECOGNITION

1.1 INTRODUCTION

Flora and fauna are the two primary ecological resources available on Earth (Cope *et al.* 2012). Among the major ecosystems in India, forest encompasses about 21.05% of the geographical area (Arisdason & Lakshminarasimhan 2016). The total number of identified plant species is around 4,65,688, of which 49,003 are of Indian origin (Source: www.bsienvis.nic.in).

The ecosystem is a complex, interconnected system, made up of biotic and abiotic constituents. All living creatures such as plants, animals, and microorganisms are biotic components. Abiotic components consist of non-living elements such as soil, water, atmosphere, etc. Flora and fauna are the most engrossing of all the biotic components. Flora is essential to human life. Humans consume plants directly or indirectly for their survival. Plants are used as food, medicines, and for the production of different products in industries (Du *et al.* 2007).

The majority of living organisms consume plants and act as primary consumers in the food chain. Two or more interconnected food chains are known as a Food Web. A simple Food Web is presented in figure 1.1. To a larger extent, the ecosystem would be affected if the plants are not protected from extinction (Rzanny *et al.* 2019).

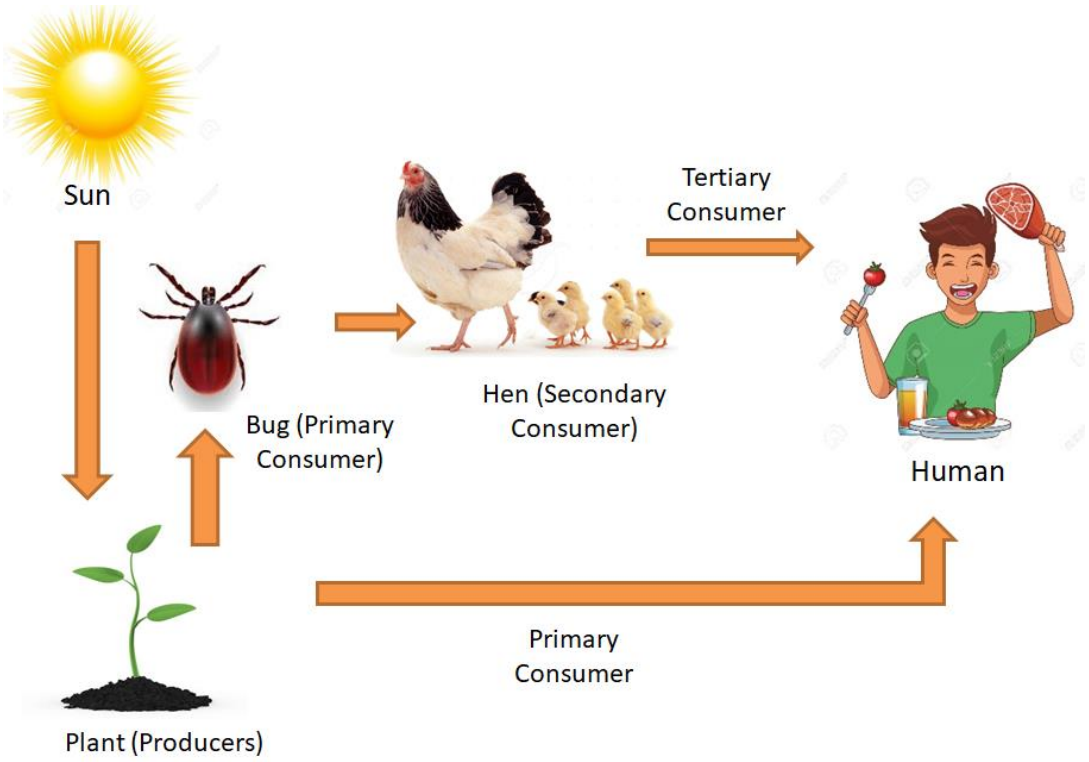


Figure 1.1 Food Web

Some of the plant species possess medicinal properties. Kala *et al.* (2006) reported that around 7500 medicinal plants are available in India. The leaves, fruits, roots, barks, or flowers of medicinal plants help to cure various health issues or diseases. Few examples of medicinal plant species are Indian Borage, Gooseberry, Wood Apple, Tulsi, Neem, Aloe Vera, etc. Tulsi helps in the treatment of fever, vomiting, liver problems, blood purification, cold, flu, asthma, and malaria (Kala *et al.* 2006). Gooseberry fruit helps alleviate the problem of constipation and diabetes (Kala *et al.* 2006). Wood apple's fruit and bark are used in the treatment of dysentery, diarrhea, and fever (Kala *et al.* 2006).

There are about 2352 ethnobotanical plants in India (Jain 1994). More than a thousand plant species are frequently used in Siddha, Ayurvedha, and Unani systems of Indian Medicine (Kader 2014). Sobitharaj (1994) carried out extensive research on plants present in Tamilnadu temples. The author

identified that those plants are called as sacred due to their medicinal values and practical uses.

In recent years, Biodiversity conservation is a major concern due to the increase in pollution, deforestation, encroachment in the form of buildings and over usage of plants. Because of the direct or indirect human activities (Murphy *et al.* 2014; Wäldchen & Mäder 2018), the available number of plant species is on a decreasing trend. With the decrease in natural resources, it is essential to preserve the plant biodiversity. To conserve the biodiversity, identification and recognition of plant species is essential. It facilitates the protection of the highly important plant species.

1.2 METHODS TO IDENTIFY THE PLANT SPECIES

Each plant species has a botanical name. The unknown plant species are identified by comparing it with the previously collected specimens. This method is known as plant identification (Valliammal 2013). When the plant name is identified, other details such as its geometry, medicinal values, nature of plantation can be obtained. Plant Classification is the process of grouping the known plant species, based on their relationship. The features such as color, shape, texture, geometry are used to categorize the plants. The plant recognition process requires an expert advice or technology (using computer vision tools).

1.2.1 Identification by an Expert

The visual identification of plant species is possible by an expert. The expert identifies the plant based on its taxonomy, herbarium, and morphology.

1.2.1.1 Plant Morphology

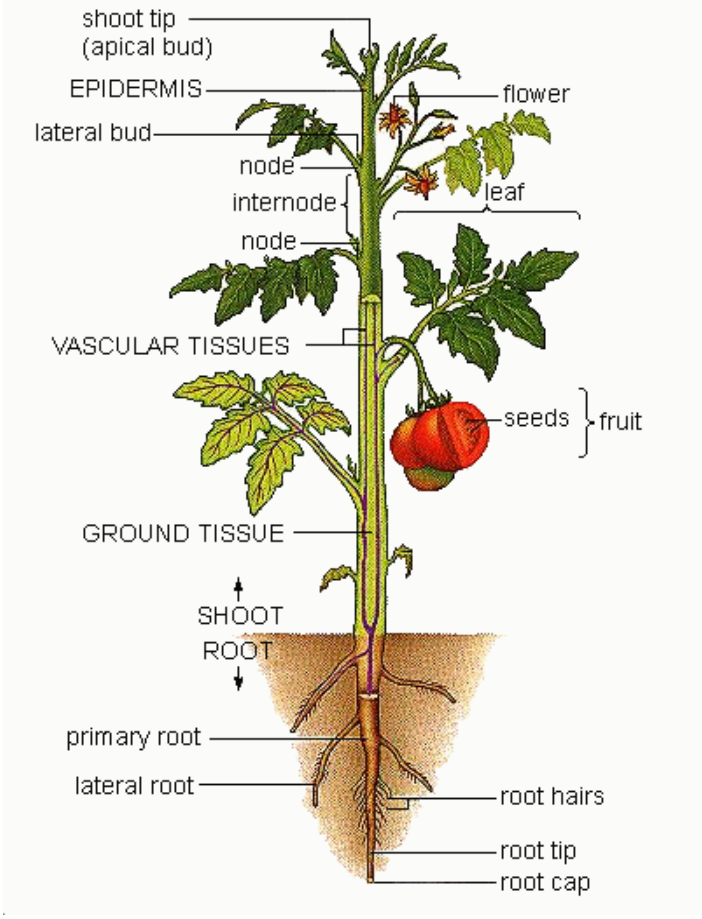


Figure 1.2 Structure of Plant System (Source: www.gitawati.wordpress.com)

Plant morphology is the study of internal and external structures in plants. It aids in the visual identification of plant species. The structure of a plant system is shown in Figure 1.2. Leaves are flattened and broad in nature (Wäldchen & Mäder 2018). The color of the leaf is due to the presence of chloroplasts (contains chlorophyll). Leaves help in the preparation of nutrients for the plants during the photosynthesis process. Flowers are the reproductive part of a plant. When they mature, they turn into fruits. Fruits are ripened ovaries containing the seeds. The root system absorbs the water and other nutrients from the soil. It acts as an anchor for the plant stem and stores the food.

Leaves play a vital role in the botanical reference, field studies compared to other parts (Le *et al.* 2014). The leaves survive in a plant for longer time than the fruits and flowers. Also, the shape of fruits, flowers, seeds possess a complex 3D structure compared to the leaves (Du *et al.* 2007). Wäldchen & Mäder (2018) indicated that around 106 studies utilized the simple plant leaves for recognition. There are about 29 research articles on plant species recognition using the compound leaves. In this thesis, the leaves (simple and compound structures) are used to identify and recognize the plant species.

1.2.1.2 Plant Taxonomy

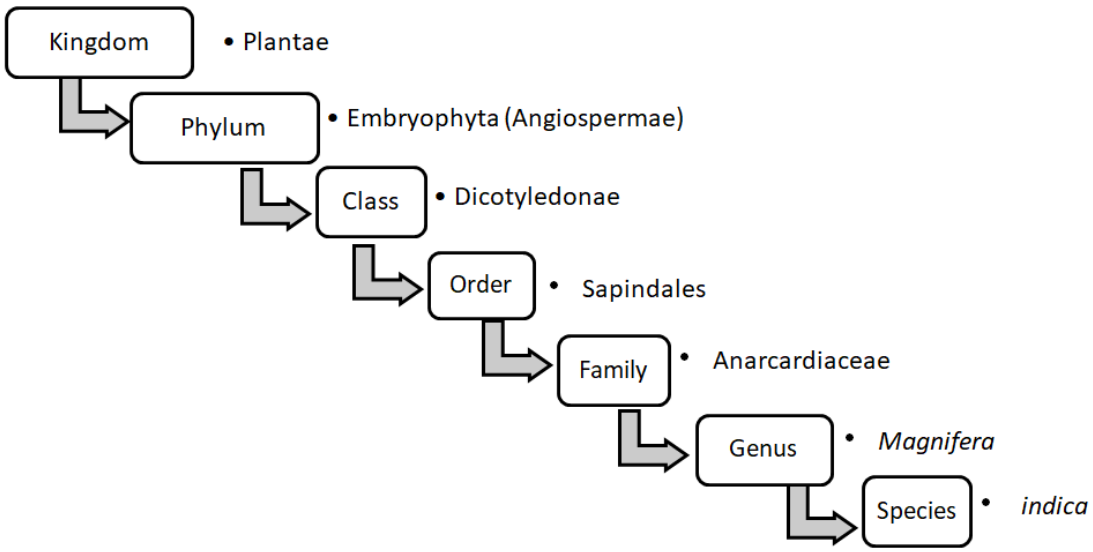


Figure 1.3 Plant Taxonomy for Mango (Source: www.sciencelearn.org.nz)

Plant taxonomy deals in identifying, describing, and classifying the plant species. It helps in the systematic arrangement of plant taxa. It aids in gathering plant information such as origin, habitat, and spatial distribution of plants. As an example, the hierarchy of plant taxonomy for mango is shown in Figure 1.3.

1.2.1.3 Herbarium for Plant Species Recognition

Herbarium (Anderson 1999; Maden 2004) is a collection of botanical specimens. Luca Ghini introduced the art of collecting and preserving the plant specimen in herbarium sheets. The collected plant specimens are dried and pressed in the blotter. Each plant specimen is placed on a separate herbarium sheet. After mounting the specimens on the herbarium sheets, each sheet is labeled with the place of its origination, family, botanical name, locality of collection, date of collection, habitat, field notes, collection number, and collector's name. Then, the herbarium sheets are arranged and maintained in a folder. Herbarium acts as a principal proof in identifying the plant species from a specific region. It also acts as a research material.

1.2.2 Plant Species Recognition using Computer Vision Tools

Previously specified methods for plant species recognition require a taxonomic expert. It is also a time-consuming process (Pang & Lim 2019). Computer vision techniques are used to ease the plant species recognition process. Computer vision is a field that requires a high-level understanding of images or videos and performs the task similar to that of the human visual system. The applications of computer vision methods include Plant species identification, Object detection, and tracking, Augmented reality, Biometrics, Character recognition, Forensics, Face recognition, Security, Surveillance, Process control, Remote sensing, Geoscience, Medical image restoration, and analysis.

The steps involved in the Image classification process is shown in Figure 1.4. It includes pre-processing, segmentation or enhancement, feature extraction, and classification. The sequence of still images or frames obtained from the video acts as an input. Usually, the images are resized or enhanced for better clarity. The feature descriptors are applied to extract the features from the

preprocessed images. Further, the classification is being performed by using Machine Learning (ML) algorithms.

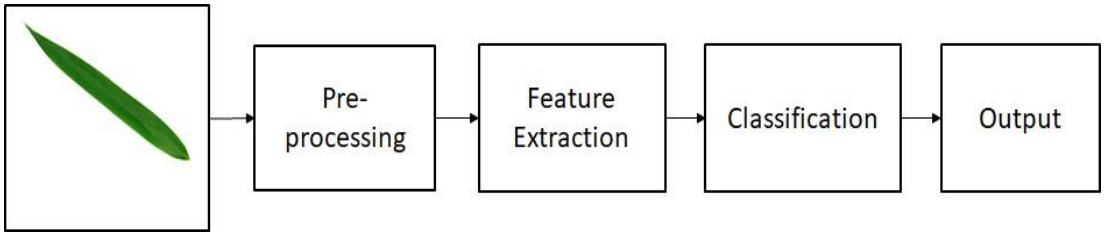


Figure 1.4 Steps in Image Processing

Deep Learning (DL) has made a breakthrough in the field of computer vision or image processing. The classification accuracies obtained by using the deep learning methods are high compared to the traditional methods of classification. The steps involved in the process of Plant Species Recognition (PSR) using the DL model in image processing is shown in Figure 1.5. Though the block diagram of the DL model looks simple, it involves mathematical operations requiring a Graphical Processing Unit (GPU) for faster computation.

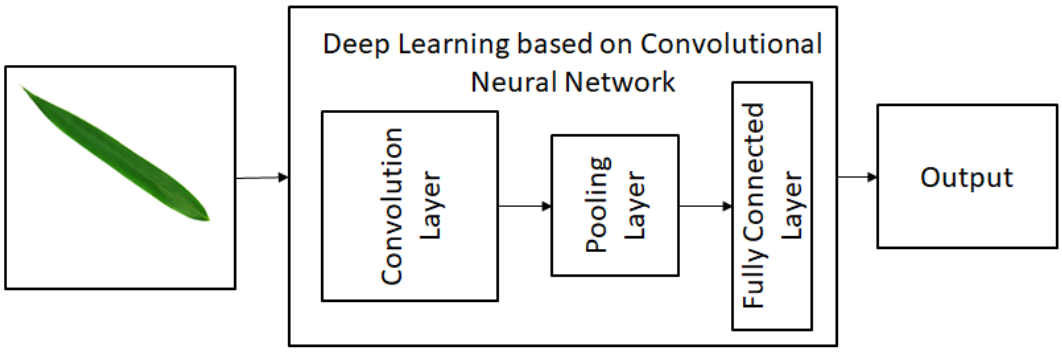


Figure 1.5 Steps in Deep Learning Process

1.3 RESEARCH MOTIVATION

Plants are available in plenty. However, people hardly recognize their names through visual perception. Due to the large diversity of plants, the role of botanist becomes inevitable in the identification of plant species. Also,

skilled taxonomists for assisting a common man with plant identification are lesser in number (Gaston & O'Neill 2004). Hence, it becomes necessary to introduce an efficient system to recognize the plant species.

This thesis focuses on plant species recognition by utilizing computer vision techniques. The plant species recognition method introduces numerous challenges with respect to the input images and leaf characteristics. Some of the challenges include the variation in illumination, scale changes, rotation, and orientation of the object. Also, plant species recognition is affected by the variation in leaf color (due to disease or seasonal changes or aging) and leaf shapes (Saleem *et al.* 2019). The challenges involved in plant species recognition based on leaf structure and leaf images are discussed in 1.3.1 and 1.3.2. Leaf-12 dataset (12 plant species of Indian origin) is collected with a motive to tackle the challenges mentioned in subsections 1.3.1 and 1.3.2.

1.3.1 Challenges involved in Plant Species Recognition due to variation in Leaf Structure and Arrangement

A simple teethed leaf and its parts are shown in Figure 1.6 (Wäldchen & Mäder 2018). The topmost tip in a leaf is called as an Apex. A single leaf is also termed as a Blade. The leaf blade is connected to the plant stem through leafstalk or petiole. The undivided leaf blade is known as a Simple leaf. The division of a leaf blade into two or more leaflets is called a compound leaf. Many leaflets are arranged in the rachis. The point that holds the leaf blade to the petiole is called as a Basepoint. The boundary of the leaf is called a Leaf margin. The types of leaf margin are discussed in subsection 1.3.1.2. Veins are the vascular tissues in the leaf’s internal structure, and it varies from one plant species to the other.

1.3.1.1 Simple or compound leaves

Figure 1.6 shows a simple and compound leaf. A simple leaf or a single leaf is connected to the stem. Compound leaf represents multiple leaves. Compound leaves are either palmately compound, pinnately compound, or bipinnately compound, as represented in Figure 1.7.

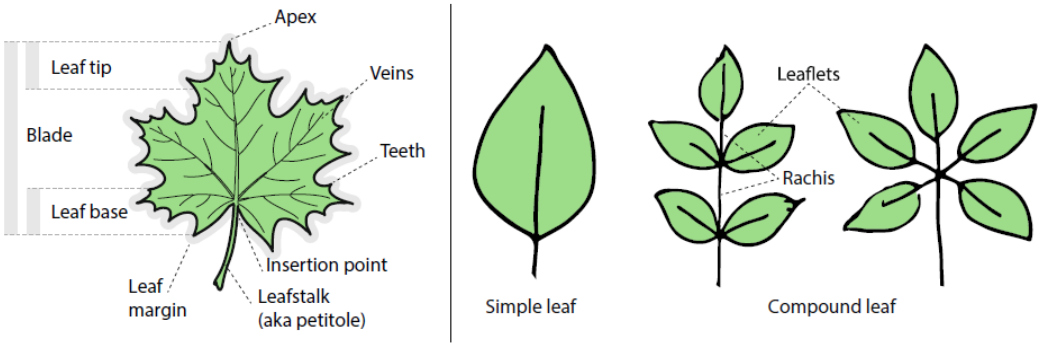


Figure 1.6 Leaf parts, Simple leaf and Compound leaf (Source: Wäldchen & Mäder 2018)

1.3.1.2 Variation in Leaf Shapes, Venation, and Margins

The shape of the plant leaves varies from one species to another. Hence, it becomes difficult to recognize the plant species based on the leaf shapes, margins, and leaf arrangement to the stem. The typical representation of leaves is shown in figure 1.7. The leaf shapes are of the form linear, obovate, ovate, pinnately lobed, palmately lobed, reniform, lanceolate, and sagittate. Leaf venation is classified as pinnate, parallel, and palmate. The different leaf margins of plant species are termed as entire, crenate, dentate, serrate, and lobed. As an example, the rose leaves possess a serrated margin (Available in Folio and Leaf-12 datasets). *Plectranthus amboinicus* (Indian Borage in Leaf-12) leaves are ovate (shape), crenate to dentate (margin) in nature. *Quercus* species is an example of lobed leaves (Available in Swedish leaf dataset).

1.3.1.3 Leaf Arrangement

Leaf arrangement on the stem varies from one plant species to another. On the stem, the leaves are arranged as either alternate, opposite or whorled. For example, the Indian borage leaves are arranged as one above the other. Whereas, the rose leaves are alternately arranged in the stem.

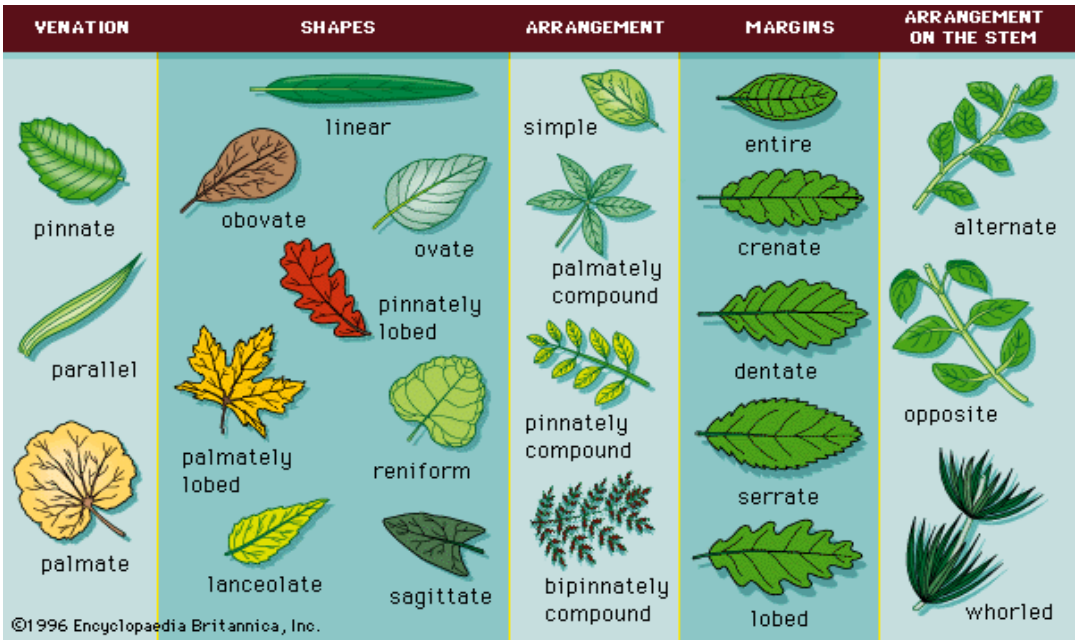


Figure 1.7 Venation, Various Shapes, Arrangement, Margins of Leaves and Leaf Arrangement on the Stem (Source: www.britannica.com)

1.3.1.4 Inter-similarity between the Plant Species

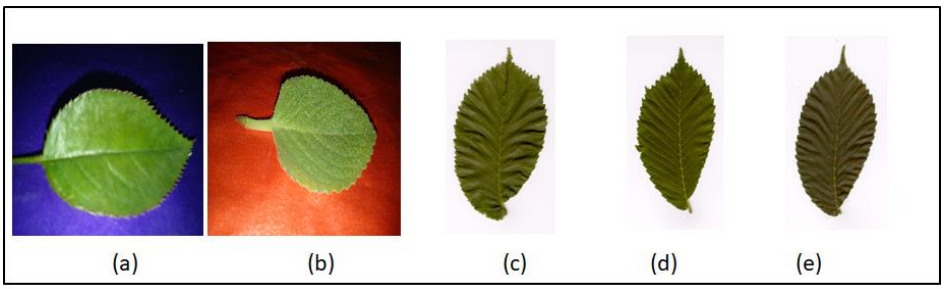


Figure 1.8 Leaves from Leaf-12 Dataset (a & b) and Swedish leaf Dataset (c, d & e)

In few scenarios, the leaf shapes of different plant species are similar in nature. It increases the chance of mispredictions. In the following figure, 1.8 (a) and (b) are rose and Indian borage leaves. Also, in figure 1.8 (c), (d), and (e), the leaves appear similar to each other. The samples are *Ulmus carpinifolia* (Figure 1.8(c)), *Salix aurita* (Figure 1.8(d)), and *Ulmus glabra* (Figure 1.8(e)) from the Swedish leaf dataset.

1.3.2 Challenges in Image-based Plant Species Recognition (PSR)

Image-based PSR framework involves numerous challenges such as illumination changes, scale changes, variation in rotation, and viewpoints. The Plant Species Recognition system (PSR) should be able to recognize the plants, irrespective of the nature of input images.

1.3.2.1 Illumination Changes

Real-time PSR system fails to recognize the plant species if the object (leaves) being examined are captured under various lighting conditions such as darkness, flashlights, etc. Some sample images are shown in Figure 1.9. In the Folio dataset (Figure 1.9 (a)-(d)), the leaf images are captured under proper illumination with white background. The color of the background appears to be varied as there is a variation in leaf color for Chocolate tree leaf (*Euphorbia cotinifolia*), as shown in Figure 1.9(b).

In the Leaf-12 dataset, the images of leaves are captured under different lighting [dim, dimmer, bright, and brighter lighting (with and without flash)] conditions. Sample images of Bottle Gourd captured under varying illumination conditions are shown in figure 1.9 (e)-(h).

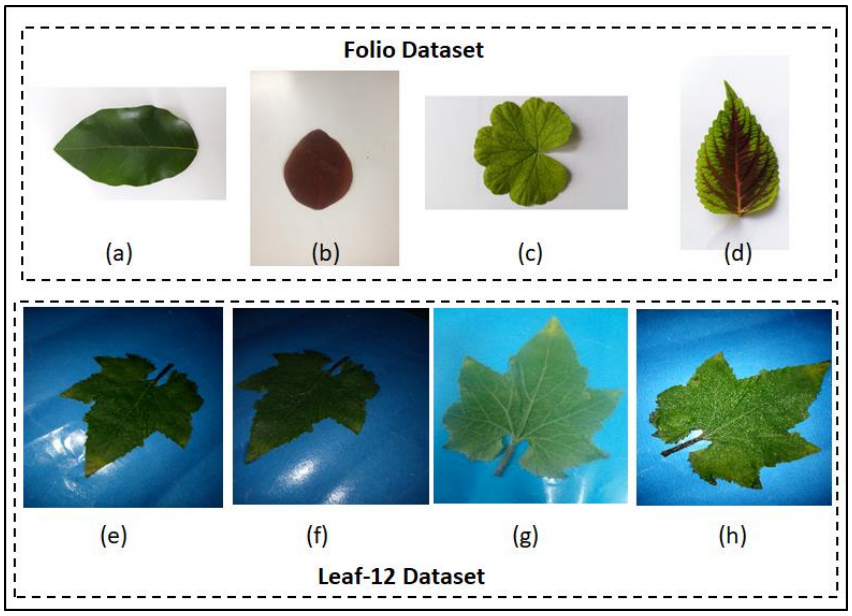


Figure 1.9 Leaf under various lighting conditions. Folio dataset (a) Barbados cherry (b) Chocolate tree (c) Geranium and (d) Vieux garcon. Dim, dimmer, bright and brighter Bottle Gourd leaf from Leaf-12 dataset are shown in (e), (f), (g) and (h)

1.3.2.2 Rotation

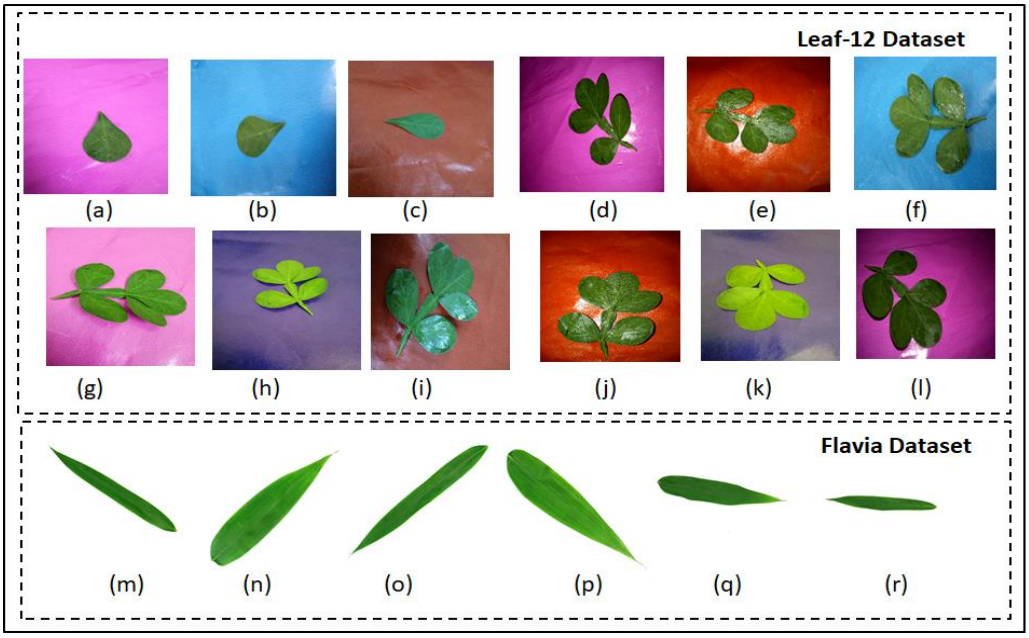


Figure 1.10 Rotated leaf samples of Wood Apple (Leaf-12 dataset) (a)-(l) and Pubescent Bamboo (Flavia dataset)–(m)-(r)

The real-time PSR framework should be able to recognize the plant species irrespective of the object's orientation or the viewpoints of the imaging device. Leaf-12 dataset images are captured from different viewpoints (top and side views), as shown in 1.10 (a)-(l). Sample images from the Flavia dataset are shown in Figure 1.10 (m)-(r).

1.3.2.3 Color Background

Images acquired with different color backgrounds possess variation in intensities of RGB channels. At times, there is a brighter visualization of leaves under a darker background and vice-versa. Sample images of the Gooseberry leaves with various color backgrounds from the Leaf-12 dataset are shown in figure 1.11 (a)-(e).

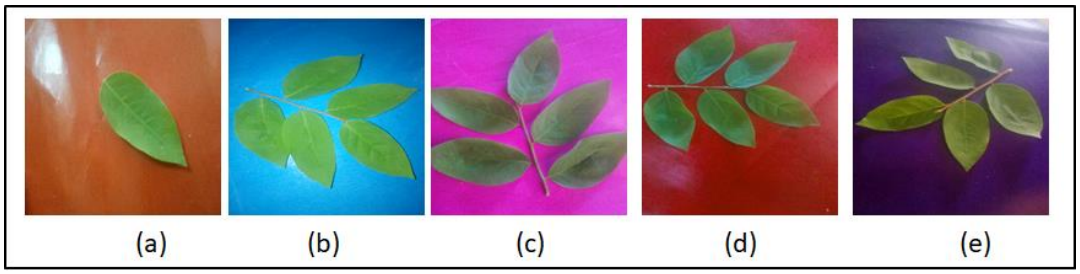


Figure 1.11 Gooseberry Leaves captured with Different Backgrounds (a) Orange (b) Blue (c) Pink (d) Red (e) Violet

1.3.3 Non-Availability of Plant Dataset of Indian Origin

There are numerous standard datasets available, including Flavia dataset, Swedish leaf, Leafsnap dataset, Folio dataset, etc. It is observed that the standard dataset contains plant species from different parts of the world, such as United States, Europe, China. The origin of the above-mentioned datasets is shown in Table 2.1 (Chapter 2-Datasets). Naresh & Nagendraswamy (2015) introduced an Indian medicinal plant dataset called UoM, collected from Mysuru, Karnataka. It has access restrictions. These datasets contain images

with single color background (pink or white). Also, these datasets do not possess the real-time image of plants.

Hence, in this research work, a self-collected real-time plant dataset of Indian origin, named as Leaf-12 is utilized. The real-time dataset consists of a single/simple leaves and compound leaves. The leaf images are captured under different lighting conditions, variations in scale, rotation or viewpoint changes, and different backgrounds.

1.4 RESEARCH OBJECTIVES AND RESEARCH CONTRIBUTIONS

The research objectives are listed below.

- To collect a plant dataset of Indian origin.
- To design and develop a PSR framework using the Conventional image processing method (feature extraction and classification).
- To analyze the performance of a Neural Network for the PSR framework. Also, to optimize the parameters for a better prediction rate.
- To experiment with the different single Deep Learning architectures for Plant Species Recognition (PSR) System.
- To design and develop a Dual Deep Learning Architecture (DDLA) for the PSR framework.

To handle the problems mentioned in section 1.3, the important research contributions reported in this thesis are listed below.

- ❖ The plant leaf dataset of Indian origin is collected and utilized. It is named as Leaf-12 dataset. It contains about 12 plant species with 320 images/plant species.
- ❖ In the Feature Extraction process of the Conventional image processing method, the fusion of Color, Shape, Texture Features

(CSTF) is carried out. The extracted features are classified using Machine Learning (ML) classifiers. The methodology is tested for the PSR.

- ❖ PSR system is tested utilizing a custom-developed Backpropagation Neural Network with the PReLU activation function.
- ❖ PSR system is analyzed by utilizing two approaches of single Deep Learning (DL) architectures, namely, DL for feature extraction + ML classifier and fine-tuned DL method.
- ❖ Dual Deep Learning Architecture is proposed for the PSR framework.

All the aforementioned contributions are evaluated using three standard datasets (Flavia (Wu *et al.* 2007), Swedish leaf (Soderkvist 2002), Folio (Munisami 2015)) and real-time dataset (Leaf-12).

1.5 SCOPE OF THE RESEARCH

Identifying and recognizing the plant species through computer vision techniques is highly advantageous and useful because of the following potential applications.

i. **Mobile/ Web Application:** Non-professional users may not be able to recognize the plant species by visualization. In such scenarios, the person may utilize a mobile or web application to identify the plant species (Joly *et al.* 2014). In mobile or web application, the user inputs the query image to retrieve the details of the plant and its usage. Such applications act as an electronic field guide for plants.

ii. **Virtual Leaf Collection:** Botanist preserve the plant species in herbarium sheets. Later, it is used to identify the plant species. However, the preserved species decay with time (Naresh 2016). Also, the

Herbarium sheets decay with time. Preserving the plant species in digital form is a viable solution to address the above-specified problem.

iii. **Preservation of endangered/endemic species:** With a decline in natural resources, it is very crucial to preserve the biodiversity of plants. Plant identification and recognition help gain information about the endangered or endemic species.

iv. **Research on Medicinal Plants:** Plant species recognition helps in identifying the medicinal plants. Medicinal plants are widely used in Siddha treatments to cure several diseases.

v. **Food and Medicinal Products:** The recognition of plant species helps to differentiate plants as food and medicinal plants. Edible Plants are used to produce food-related products. Plants are used in the preparation of natural medicines to treat ailments.

1.6 THESIS STRUCTURE

The research workflow of this dissertation on plant species recognition is shown in Figure 1.12. **Chapter 1** discusses the plants and their importance, different methods, and challenges involved in plant species recognition. The scope and contributions of the research are also described. **Chapter 2** details the plant datasets (standard and custom developed) used in this thesis for the experimental evaluation.

Chapter 3 deals with the performance analysis of the PSR framework based on Conventional image processing methods. **Chapter 4** covers the classification of plant species using custom-developed Backpropagation Neural Network with Parametric Rectified linear unit (PReLU) activation function.

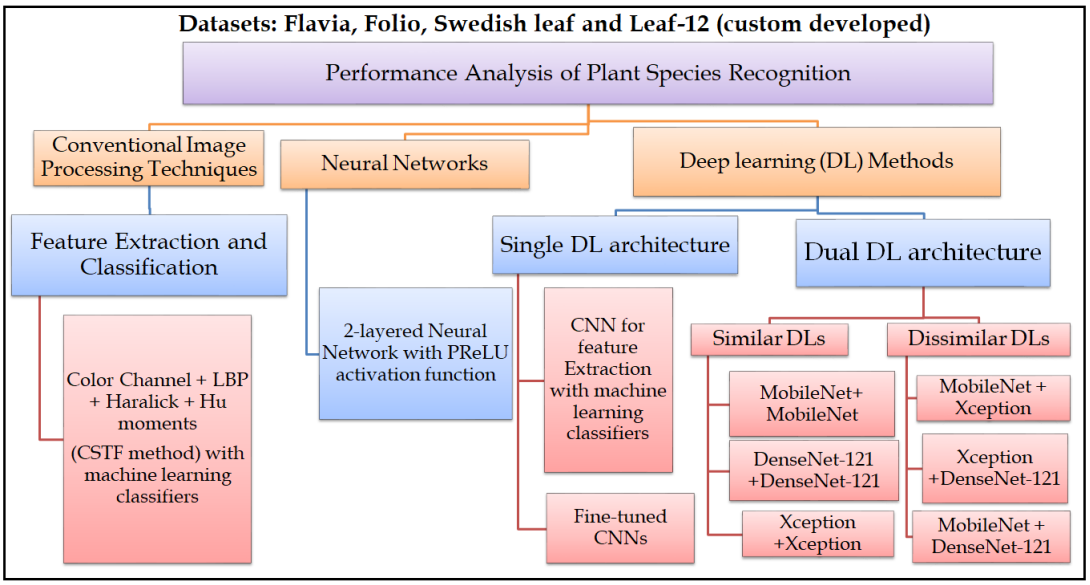


Figure 1.12 Research Workflow on Plant Species Recognition

Chapter 5 discusses the performance analysis of several single Deep Learning architectures for plant species recognition. **Chapter 6** reports the usage of Dual Deep Learning Architecture (DDLA) in the context of the PSR framework. **Chapter 7** summarizes the results of this research work.

CHAPTER 2

PLANT DATASETS

2.1 AVAILABILITY OF PLANT DATASETS

Datasets play an important role in plant species recognition. The quantity, quality, and availability of datasets determine the success story of classification algorithms. Numerous plant datasets are available. Few plant datasets consist of flowers or leaves as different classes. Since plant leaves are available throughout the year, it is widely used in plant species recognition (Aakif & Khan 2015). Hence, in this dissertation, leaf-based plant species recognition is carried out. The available plant leaf datasets are mentioned in Table 2.1.

Table 2.1 List of Plant Datasets

S. No.	Plant Datasets	Plant Part	Open Source	Collected from (place)	Author & Year
1	UCI Plant Leaf Dataset	Leaf	Yes	Porto, Portugal	Silva *et al.* 2013
2	UCI One Hundred Plant Species Dataset	Leaf	Yes	Royal Botanical Garden, Kew, UK	Mallah *et al.* 2013
3	LeafSnap Dataset	Leaf	Yes	North-Eastern United States and Canada	Kumar *et al.* 2012
4	Flavia Dataset	Leaf	Yes	Nanking, China	Wu *et al.* 2007
5	Folio Dataset	Leaf	Yes	Mauritius	Munisami *et al.* 2015
6	Swedish leaf Dataset	Leaf	Yes	Sweden	Soderkvist 2002

LeafSnap dataset consists of 183 classes of plant species. Each class contains about 11 to 145 images. UCI Plant Leaf Dataset consists of 40 classes with 5 to 13 images per class. UCI One Hundred Plant Species Dataset

contains about 100 different classes, and each class has about 16 binarized image samples. From table 2.1, it is inferred that there is no open access standard dataset for plant species of Indian origin.

Three standard datasets (Flavia, Swedish leaf, and Folio) are chosen based on either frequency of citation or a reasonable number of images in the dataset. Other datasets mentioned in Table 2.1 contain images with large variations in images per class. Two standard leaf datasets are frequently used by numerous research groups, namely, Flavia (Wu *et al.* 2007) and Swedish leaf (Soderkvist 2002**)** datasets for estimating the performance of their methods. Also, a dataset collected from the UCI machine-learning repository, named Folio (Munisami *et al.* 2015), is used to estimate the performance metrics. In addition, a self-collected real-time dataset from Indian plant species known as 'Leaf-12' is used. The dataset related information is presented in subsections 2.1.1-2.1.4 of this chapter. The image reconstruction method is discussed in subsection 2.2.

2.1.1 Flavia dataset

Flavia dataset (Wu *et al.* 2007) is a benchmark dataset. Wu *et al.* (2007) have collected images of 32 different plant species. Altogether, 1907 images are available in the dataset for 32 species. In this research work, 50 images per class are utilized (1600 images for 32 classes). Sample leaf images of the Flavia dataset are shown in Figure 2.1, and their botanical names are listed in Table 2.2. All the leaves in the Flavia dataset are simple leaves with a white background. The images in this dataset are with variation in orientation and lighting conditions.

Flavia dataset is used for both Deep Learning (DL) and Non-DL approaches, as mentioned in Table 2.3. Some of the DL approaches that use the Flavia dataset are Convolutional-Recurrent Neural Networks (C-RNNs) (Liu *et*

al. 2018), five-layered CNN (Thanh *et al.* 2018) and transfer learning-based deep networks (Kaya *et al.* 2019).

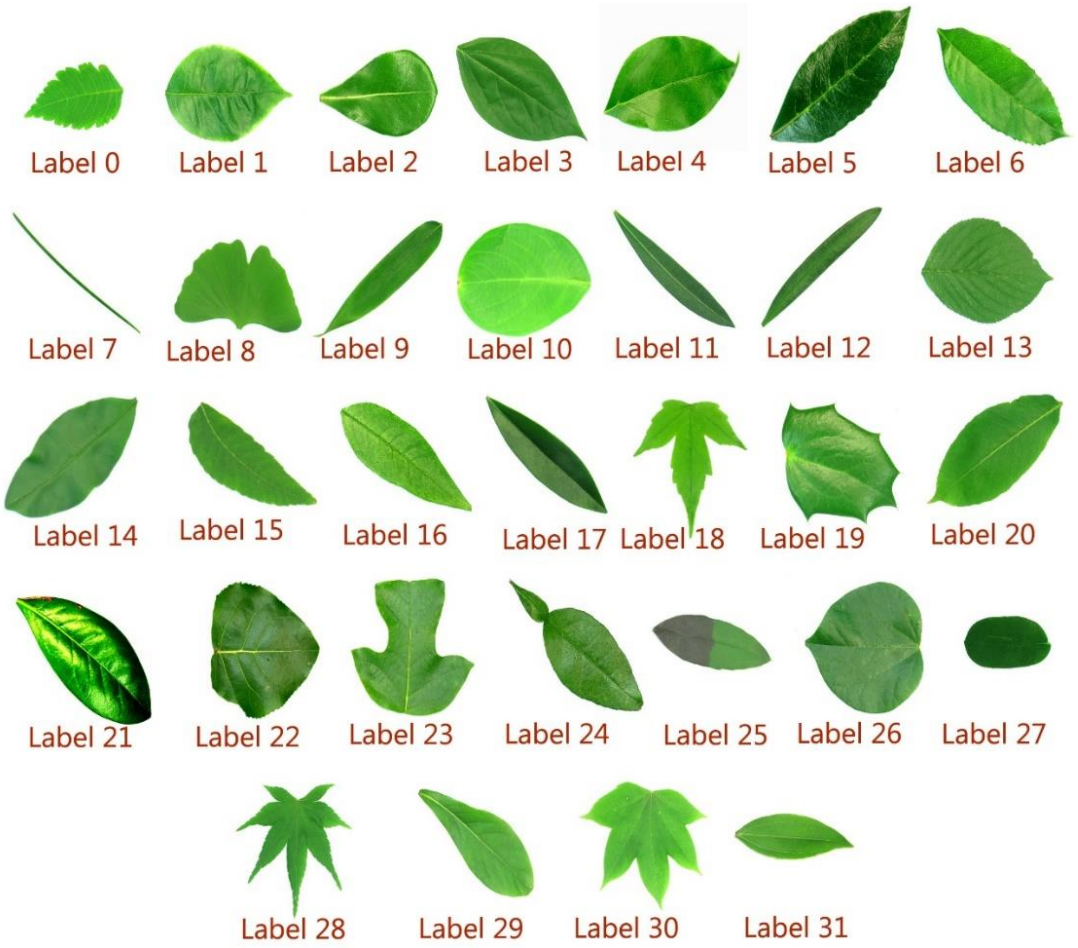


Figure 2.1 Sample leaf images of Flavia Dataset (Wu *et al.* 2007)

Table 2.2 List of Plant Names from Flavia Dataset (Wu *et al.* 2007)

Label	Botanical Name	Common Name
0	*Koelreuteria paniculata Laxm.*	Goldenrain Tree
1	*Ilex macrocarpa Oliv.*	Big-Fruited Holly
2	*Pittosporum tobira (Thunb.) Ait. f.*	Japanese Cheesewood
3	*Chimonanthus praecox L.*	Wintersweet
4	*Cinnamomum camphora (L.) J. Presl*	Camphor Tree
5	*Viburnum awabuki K.Koch*	Japan Arrowwood
6	*Osmanthus fragrans Lour.*	Sweet Osmanthus
7	*Cedrus deodara (Roxb.) G. Don*	Deodar
8	*Ginkgo biloba L.*	Ginkgo, Maidenhair Tree

Table 2.2 (Continued)

9	*Phyllostachys edulis (Carr.) Houz.*	Pubescent Bamboo
10	*Lagerstroemia indica (L.) Pers.*	Crape Myrtle, Crepe Myrtle
11	*Nerium oleander L.*	Oleander
12	*Podocarpus macrophyllus (Thunb.) Sweet*	Yew Plum Pine
13	*Prunus serrulata Lindl. var. lannesiana auct.*	Japanese Flowering Cherry
14	*Ligustrum lucidum Ait. f.*	Glossy Privet
15	*Tonna sinensis M. Roem.*	Chinese Toon
16	*Prunus persica (L.) Batsch*	Peach
17	*Manglietia fordiana Oliv.*	Ford Woodlotus
18	*Acer buergerianum Miq.*	Trident Maple
19	*Mahonia bealei (Fortune) Carr.*	Beale's Barberry
20	*Aesculus chinensis*	Chinese Horse Chestnut
21	*Magnolia grandiflora L.*	Southern Magnolia
22	*Populus ×canadensis Moench*	Canadian Poplar
23	*Liriodendron chinense (Hemsl.) Sarg.*	Chinese Tulip Tree
24	*Citrus reticulata Blanco*	Tangerine
25	*Berberis anhweiensis Ahrendt*	Anhui Barberry
26	*Cercis chinensis*	Chinese Redbud
27	*Indigofera tinctoria L.*	True Indigo
28	*Acer Palmatum*	Japanese Maple
29	*Phoebe nanmu (Oliv.) Gamble*	Nanmu
30	*Kalopanax septemlobus (Thunb. ex A.Murr.) Koidz.*	Castor Aralia
31	*Cinnamomum japonicum Sieb.*	Chinese Cinnamon

Currently, the reported literatures that utilize the Flavia dataset for Non-DL approaches are Multi-class Twin Support Vector Machine (Goyal *et al.* 2019), conventional method (Saleem *et al.* 2019), a combination of CUR decomposition and Weighted Kernel Sparse Representation (CUR-WKSR) (Pankaja & Suma 2019). Also, a Non-DL approach using Local Binary Pattern (LBP) based descriptors such as Region Mean-LBP, Overall Mean-LBP, and ROM-LBP (Region Overall Mean LBP) (Turkoglu & Hanbay 2019) are reported.

Table 2.3 Recent Studies on Flavia Dataset

DL/ Non-DL Approach	Method	Recent Studies
DL	C-RNN	Liu *et al.* (2018)
DL	five-layered CNN	Thanh *et al.* (2018)
DL	Custom created CNN	Kaya *et al.* (2019)
	AlexNet + SVM, AlexNet + LDA	
	VGG-16 + SVM, VGG-16 + LDA	
	Fine-tuned AlexNet and VGG-16	
	AlexNet + RNN, VGG-16 + RNN	
Non-DL	Multi-class Twin Support Vector Machine	Goyal *et al.* (2019)
Non-DL	Conventional Method	Saleem *et al.* (2019)
Non-DL	LBP based Descriptors	Turkoglu & Hanbay (2019)
Non-DL	CUR-WKSR	Pankaja & Suma (2019)

2.1.2 Swedish Leaf dataset

Figure 2.2 Swedish leaf Dataset (Soderkvist 2002)

Swedish leaf dataset (Soderkvist 2002) contains about 15 different leaves of plants. Each class in the dataset contains about 75 images. Figure 2.2 shows the sample image of each leaf in the Swedish leaf dataset. The botanical names and common names are mentioned in Table 2.4. This dataset includes inter-similar plant species.

Table 2.4 List of Plant Species in Swedish leaf Dataset (Soderkvist 2002)

Label	Botanical Name	Common Name
0	*Ulmus carpinifolia*	-
1	*Sorbus aucuparia*	Mountain ash
2	*Salix sinerea*	Common sallow
3	*Populus*	-
4	*Tilia*	-
5	*Sorbus intermedia*	Swedish whitebeam
6	*Fagus silvatica*	European beech
7	*Acer*	-
8	*Salix aurita*	Eared willow
9	*Quercus*	-
10	*Alnus incana*	Grey alder
11	*Betula pubescens*	Downy birch
12	*Salix alba 'Sericea'*	Silver willow
13	*Populus tremula*	Common aspen
14	*Ulmus glabra*	Scotch elm

Table 2.5 shows the recent literatures utilizing the Swedish leaf dataset in estimating the performance of the reported methods. In Non-DL approaches, a combination of two algorithms (Bag of Words (BoW) and Entropy Sequence (EnS) (Wang *et al.* 2018)) and multi-class Support Vector Machine (Kaur &Kaur 2019) is reported.

Table 2.5 Recent Studies on Swedish leaf Dataset

DL/ Non-DL Approach	Method	Recent Studies
DL	CNN	Atabay (2016)
DL	CNN	Thanh *et al.* (2018)
DL	Custom created CNN	Kaya *et al.* (2019)
	AlexNet + SVM, AlexNet + LDA	
	VGG-16 + SVM, VGG-16 + LDA	
	Fine-tuned AlexNet and VGG-16	
	AlexNet + RNN, VGG-16 + RNN	
Non-DL	BoW and EnS	Wang *et al.* (2018)
Non-DL	Multi-class Support Vector Machine	Kaur &Kaur (2019)
Non-DL	CUR-WKSR	Pankaja & Suma (2019)
Non-DL	LBP based descriptors	Turkoglu & Hanbay (2019)

Few more Non-DL plant recognition methods such as CUR-WKSR (Pankaja & Suma 2019) and LBP based descriptors (Turkoglu & Hanbay 2019) utilize the Swedish leaf dataset. Atabay (2016) and Thanh *et al.* (2018) proposed a Convolutional Neural Network (CNN) model for the classification of plant species. Using this dataset, Kaya *et al.* (2019) assessed their four transfer learning-based CNN models.

2.1.3 Folio Dataset

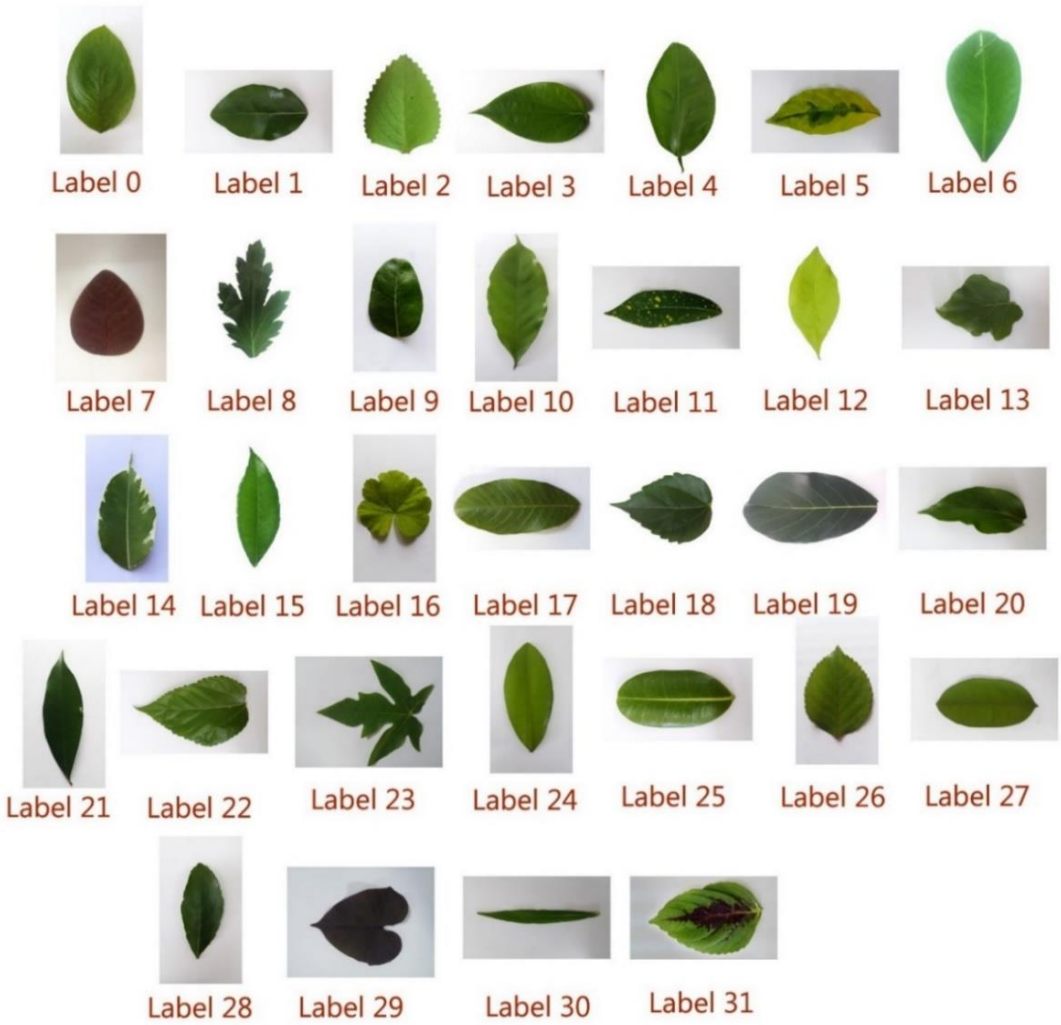


Figure 2.3 Samples of Folio Dataset (Munisami *et al.* 2015)

Folio dataset (Munisami *et al.* 2015) is available in the UCI repository with 32 different plant species. The dataset consists of a fewer

number of images/class containing only 637 samples compared to other datasets. The results are recorded for 576 images, i.e., 18 images are used in each class. Sample images are shown in Figure 2.3.

Images in the Folio dataset (Munisami *et al.* 2015) are photographed either in the horizontal or vertical position of the leaf with a white background. The names of plant species in the Folio dataset are listed in Table 2.6.

Table 2.6 List of Plant Species in Folio Dataset (Munisami *et al.* 2015)

Label	Botanical Name	Common Name
0	*Mussaenda philippica*	Ashanti Blood
1	*Malpighia glabra*	Barbados Cherry
2	*Myroxylon balsamum*	Beaumier Du Perou
3	*Piper betle*	Betel
4	*Citrus aurantium*	Bitter Orange
5	*Graptophyllum pictum*	Caricature Plant
6	*Psidium cattleianum*	Chinese Guava
7	*Euphorbia cotinifolia*	Chocolate Tree
8	*Dendranthema grandiflora*	Chrysanthemum
9	*Carissa carandas*	Coeur Demoiselle
10	*Coffea arabica*	Coffee
11	*Codiaeum variegatum*	Croton
12	*Duranta erecta*	Duranta Gold
13	*Solanum melongena*	Eggplant
14	*Ficus benjamina*	Ficus
15	*Spondias dulcis*	Fruitcitere
16	*Pelargonium*	Geranium
17	*Psidium guajava*	Guava
18	*Hibiscus rosa-sinensis*	Hibiscus
19	*Artocarpus heterophyllus*	Jackfruit
20	*Dovyalis hebecarpa*	Ketembilla
21	*Litchi chinensis*	Lychee
22	*Morus alba*	Mulberry Leaf
23	*Carica papaya*	Papaya
24	*Pimenta dioica*	Pimento
25	*Mimusops coriacea*	Pomme Jacquot
26	*Rosa*	Rose
27	*Chrysophyllum cainito*	Star Apple
28	*Osmanthus fragrans*	Sweet Olive
29	*Ipomoea batatas*	Sweet Potato
30	*Cascabela thevetia*	Thevetia
31	*Plectranthus scutellarioides*	Vieux Garcon

The Folio dataset is used in DL as well as Non-DL approaches. It is reported in Table 2.7. Pawara *et al.* (2017a) utilized the Folio dataset for DL (AlexNet and GoogleNet), and non-DL (Histogram of Gradients-HOG, BoW) approaches. Pawara *et al.* (2017b) also utilized the Folio dataset for DL approach (scratch and fine-tuned AlexNet as well as GoogleNet for augmented

Table 2.7 Studies using Folio leaf Dataset

Non-DL/DL Approach	Method	Recent Studies
Non-DL	HOG, BoW	Pawara *et al.* (2017a)
DL	AlexNet and GoogleNet	
DL	AlexNet	Pawara *et al.* (2017b)
Non-DL	Visual features	Yigit *et al.* (2019)
Non-DL	LAII Shape Descriptor	Hewitt & Mahmoud (2018)

dataset). Yigit *et al.* (2019) (Visual features) and Hewitt & Mahmoud (2018) (Local Area Integral Invariant (LAII) Shape descriptor) used Folio dataset for Non-DL approaches.

2.1.4 Custom Dataset (Leaf-12)

Figure 2.4 Samples of Leaf-12 Dataset

Leaf-12 is a custom collected dataset with plant leaves of Indian origin. The images are captured using HTC desire-828 mobile phone (13

Megapixel camera). This dataset consists of 12 classes. The plant names are reported in Table 2.8, and the sample images are shown in Figure 2.4. The dataset images are photographed under various color backgrounds, different viewpoints, and various lighting conditions (mobile flash, shadowed leaf images). Simple leaf and compound leaves are also considered during the collection phase of the dataset. The Leaf-12 dataset consists of leaves from ornamental plants and medicinal plants.

Table 2.8 List of Plant Species in Leaf-12 Dataset

Label	Botanical Name	Common Name
0	*Lagenaria siceraria*	Bottle Gourd
1	*Citrus medica Linn.*	Citron
2	*Abutilon indicum*	Indian Abutilon
3	*Plectranthus amboinicus*	Indian Borage
4	*Phyllanthus emblica*	Indian Gooseberry
5	*Ixora coccinea*	Jungle Flame
6	*Artocarpus heterophyllus*	Jackfruit
7	*Jasminum officinale*	Jasmine
8	*Rosa*	Rose
9	*Annona reticulate*	Sugar Apple
10	*Solanum torvum*	Turkey Berry
11	*Aegle marmelos*	Wood Apple

2.2 DATASET IMAGE RECONSTRUCTION

Images in the four datasets (Flavia, Folio, Swedish leaf, and Leaf-12) discussed are resized and reconstructed using the algorithm 2.1. These reconstructed images are used in the evaluation of algorithms. The obtained results are reported in chapters 3-6. Thumbnail function resizes the image to the maximum input dimension, maintaining the aspect ratio to form (Rsz_w, Rsz_h).

Algorithm 2.1: Image Reconstruction

Input: Image, I (I_w, I_h)

Output: Reconstructed Image, Rec (Rec_w, Rec_h)

Step 1: Read Input image, I

Step 2: Resize the image with dimensions (Rsz_w, Rsz_h) along with antialiasing operation

Step 2: Calculate offset for width and height

$$offset_w \leftarrow max\left(\left(\frac{I_w - Rsz_w}{2}\right), 0\right)$$

$$offset_h \leftarrow max\left(\left(\frac{I_h - Rsz_h}{2}\right), 0\right)$$

Step 3: Place *Rsz* image after (*offset_w*, *offset_y*) pixel positions to form a reconstructed image Rec with dimensions (Rec_w, Rec_h)

The thumbnail function utilizes the Antialiasing operation. It preserves the quality of the image. The image reconstruction is carried out using the Pillow package in python. The final reconstructed image, Rec (Rec_w, Rec_h), is of size 300x300 pixels.

2.3 SUMMARY

In this chapter, the four datasets used in the experimental analysis are discussed. A custom collected dataset named 'Leaf-12' is also reported. The plant names and sample leaf images from each of the datasets are listed. Among the three standard datasets used for evaluation, it is noticed that the Flavia and Swedish leaf datasets are frequently utilized compared to the Folio dataset. Also, the image reconstruction procedure is detailed. The reconstructed plant leaf images are used in the performance analysis of different methods, as reported in the forthcoming thesis chapters.

CHAPTER 3

PERFORMANCE ANALYSIS OF PLANT SPECIES RECOGNITION USING CONVENTIONAL METHODS

3.1 INTRODUCTION

Plant Species Recognition (PSR) using the Conventional method involves image preprocessing, feature extraction, and classification, as shown in figure 1.1 (Chapter 1). As a preprocessing step, resizing of images is carried out. Then, the feature descriptors are applied to extract the features. Figure 3.1 shows the type of features (Shape, Texture, Color) extracted from the leaf images. The extracted features are classified using Machine Learning (ML) algorithms.

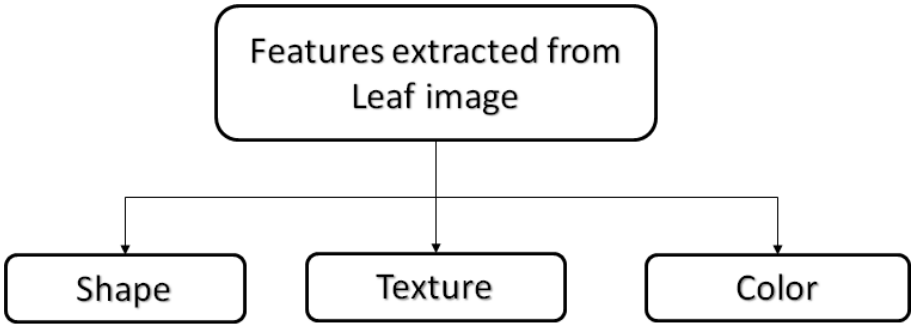


Figure 3.1 Extracted Features from Leaf Images

This chapter reports the performance of Conventional Plant Species Recognition methods. Section 3.2 discusses the reported literatures on the PSR system using Conventional techniques. Section 3.3 describes the proposed Color, Shape, and Texture Fusion (CSTF) method for the feature extraction process. Section 3.4 discusses the results obtained by using Conventional methods. Finally, Section 3.5 summarizes the outcomes of this chapter.

3.2 LITERATURE SURVEY- CONVENTIONAL METHODS

Wang *et al.* (2020) proposed a counting based leaf descriptor for plant species recognition. The author converted the color leaf images to grayscale images. The grayscale images are convoluted using Elliptical Half Gabor Wavelet to extract the line responses. Maximum Gap Local Line Direction Patterns (MGLLDP) are extracted and normalized using the line responses to obtain the normalized histogram patterns. Then, these patterns are classified using the Support Vector Machine (SVM) classifier. The method is evaluated on Swedish leaf, Flavia, and ICL datasets.

Kheirkhah & Asghari (2019) classified the plant species using GIST texture features. A pre-filtered image is created using Discrete Fourier Transform (DFT) filter and Local Contrast Normalization. GIST texture features are extracted from the pre-filtered image. Several Gabor filters are convolved with the image (color channels R, G, B) in different directions and scales. Important features are obtained by using Principal Component Analysis (PCA). Then, the selected features are classified using three classifiers, namely, Patternnet Neural Network, Support Vector Machine (SVM), and K-Nearest Neighbor (K-NN). It is reported that the performance of cosine K-NN utilizing the GIST features with PCA is superior to other methods. The methodology is tested on three datasets, namely, ImageCLEF, LeafSnap, and Flavia.

Saleem *et al.* (2019) proposed a plant species recognition system based on morphological, geometrical, and texture features. Leaf images are preprocessed and segmented to obtain the region of interest. From the segmented images, 11 shape features, 7 statistical features, and 5 venation features are extracted and normalized. The redundant features are eliminated. Ten important features are selected by utilizing the Principal Component Analysis (PCA) technique. The selected features are classified using four algorithms, such as K-NN, Decision Tree, Naïve Bayes, and multi-Support

Vector Machine (multi-SVM). K-NN classifier performed better than the other algorithms. The methodology is tested on Flavia dataset (1600 images) and a real-time self-collected dataset (625 images). The author's work is limited to mobile applications.

Zhang *et al.* (2018) proposed a combined method for plant species recognition. It comprises of Sparse Representation (SR) and Singular Value Decomposition (SVD) methods. SR technique reduces the computational complexity. The SVD method minimizes the number of features. The recognition system does not utilize any feature extraction method. Leaf images are used for creating a dictionary. The class-specific dictionary learning method is used for sparse modeling. The samples are tested with the sparse coefficients. The combined method is tested on two leaf datasets, namely, ICL leaf and Swedish leaf datasets.

Kan *et al.* (2017) proposed a system for classifying the medicinal plants from Anhui University, China. Leaf images are pre-processed by segmenting the grayscale images. Ten Shape Features (SF) are extracted. It includes three geometrical features (Eccentricity, Rectangularity, and Circularity) and seven Hu Moments. Grayscale Co-Occurrence Matrix (GLCM) based extraction of five Texture Features (TF), such as Energy, Entropy, Contrast, Correlation, and Inverse Difference Moment is carried out. Extracted features are used to classify (SVM with Radial Basis Function (RBF) kernel) the plant species. A combination of shape and texture features resulted in better classification accuracy compared to individual feature-based classification. The authors reported an accuracy of 93.3% with a low computational time of 0.0025s.

Zhang *et al.* (2017) proposed a two-stage Local Similarity-based Classification Learning (LSCL) algorithm for plant species recognition. Leaf images are converted to grayscale images. A textured image is obtained by

using the Canny edge detection algorithm. The textured image is cropped and normalized to form a feature vector of dimension 1024-d. LSCL classification algorithm is applied to the normalized feature vectors. LSCL combines the Local Mean based Clustering (LMC) and Local Sparse Representation methods. As a first step, the Euclidean distance is computed. It is used to find the nearest neighbor subsets that are proximal to the test samples. In the second step, the test sample is represented as a linear combination of all the training samples from the nearest neighbor subsets. Later, the residue is used to classify the test samples. The performance of the PSR system is evaluated on the ICL leaf dataset. The reported methodology achieved an accuracy of 0.8412±0.168 and a low computational cost of 74 s.

Carranza-Rojas & Mata-Montera (2016) presented the classification of tree species using shape and texture features. Leaf images are converted to HSV color space. Saturation (S) and value (V) channels from the HSV color model are used as inputs to the Expectation-Maximization (EM) algorithm for segmenting the leaves. Then, the images are improved by performing clipping and resizing operations. Leaf margins are identified using Histogram of Curvature over Scale (HCoS), adopted from LeafSnap's algorithm. Texture features are extracted using the Local Binary Pattern Variance (LBPV) method. K-NN algorithm is used for plant classification. The plant species recognition system resulted in an accuracy of 63% for a 'k' value of 7.

Mzhougi *et al.* (2016) categorized the leaf images as simple, compound, and lobed, rather than focusing on the classification of plant species. The histogram descriptors such as Fourier histogram, Hough histogram, Edge Orientation histogram, and Local Edge Orientation histogram are used for studying the texture and vein features. For extracting the leaf margin related information, the Directional Fragment histogram is used. The training set is categorized as simple and compound leaves. Further, the simple leaves are separated as Apical, Basal, and Middle, while the compound leaves

are separated based on leaflets. Both the simple and compound leaves follow unique indexing. In simple leaf, the fusion step is carried out by utilizing the partial similarity distances of Apical, Basal, and Middle portions. For compound leaves, the ranking list of leaflets is concatenated. The ranking list of simple and compound leaves are fused. Later, it is classified using a K-NN classifier. ImageCLEF2011 dataset is used for evaluating the method.

Wang *et al.* (2015) proposed a shape descriptor termed as Multiscale Arch Height (MARCH). For each contour point, a hierarchical arch height feature is extracted. MARCH is acquired from K-Scale Arch Height, which is a measure of curve around the contour point. For mobile retrieval of leaf images, L1-norm dissimilarity measurement is used. The shape descriptor is evaluated on Swedish leaf, Flavia, ICL, and a subset of ImageCLEF leaf datasets. An android mobile application is developed. The authors quoted that the methodology is 500 times faster than the state-of-the-art methods in image retrieval.

Zhao *et al.* (2015) introduced a counting based shape descriptor to detect the simple and compound leaves. Mixture-of-Gaussian is used in the process of leaf segmentation. After segmentation, an Independent-Inner Distance Shape Context (I-IDSC) is applied. It is based on inner distance and inner angle measurements. The reported method is used to count the number of shape patterns rather than shape features. Using Nearest Neighbor classifier, the pattern counts are determined. It is used to identify the similarity between the query and the training image. A Chi-square distance metric is employed to determine the similarity score. Five datasets, namely, Swedish leaf, ICL, Smithsonian, Plumbers Island, and custom-created dataset, are used for evaluating the I-IDSC method.

Kalyoncu & Toygar (2015) presented an integrated set of features to classify the leaves. The preprocessing steps, such as segmentation, noise

removal, contour extraction, and corner detection, are performed. Then, the feature descriptors are used to extract the Moment Invariants, Convexity, Perimeter ratio, Multi-scale Distance Matrix (MDM), average Margin Distance, and Margin Statistics. The classification is performed by Linear Discriminant Classifier for the extracted features. The reported methodology is highly effective compared to the geometric approaches, in terms of accuracy (~95%) and computational time (356 s). The limitation of the methodology is that the leaves with identical shapes and varied textures resulted in a large number of mispredictions.

Ghasab *et al.* (2015) reported the utilization of Ant Colony Optimization (ACO) in plant species recognition. Shape, morphology, texture, and color features are extracted from the grayscale images of leaves. The extracted features are optimally selected using the ACO algorithm. Later, it is classified using SVM. The methodology is evaluated on Ficus deltoidea, Citrus, Acanthaceae (FCA) dataset (using six features), and Flavia dataset (using eleven features).

Wang *et al.* (2014) proposed a Dual-Scale Local Binary Pattern (DS-LBP) method for feature extraction. The leaf images are converted to gray-level images and normalized. The Dual-Scale decomposition method involves an adaptive lifting wavelet scheme followed by variable-scale Gaussian filters. It results in a sub-band of output images. DS-LBP based shape and texture features form the resultant sub-band images. For every sub-band image, a histogram is computed. The extracted features are classified using the Fuzzy k-Nearest Neighbor method. The performance of the methodology is evaluated over two benchmark datasets, namely, the Swedish leaf database and the ICL database.

Larese *et al.* (2014) proposed a method for leaf recognition using the multiscale features of leaf veins. This method utilizes an Unconstrained Hit and

Miss Transform (UHMT) followed by adaptive thresholding for segmenting the leaf veins (features) at various scales. Feature reduction is performed by utilizing the Recursive Feature Elimination method. The features are classified using Random Forest Classifier (RFC), Support Vector Machine (SVM) with Gaussian kernel, and Penalized Discriminant Analysis (PDA) techniques. This method is aimed at solving the problem of inter-similarity in three legume species and soybean cultivars.

Cerutti *et al.* (2013) proposed a method for the identification of tree leaves in a complex natural background. Based on the polygon leaf model and active contour segmentation algorithm, the leaf contours are obtained. The features are extracted by utilizing different models such as the Polygonal Model, Base and Tip models (B/T), and Curvature-Scale histogram (HCS). The similarity between the leaves is estimated by utilizing the Mahalanobis distance metrics.

Novotny & Suk (2013) developed a web application for the recognition of plant species in Europe. As a pre-processing step, the gray level image is thresholded to form a binary image using the Otsu thresholding method. The longest boundary in the binary image is traced as a contour. The features extracted are image moments, Fourier descriptors, and leaf size. Later, these features are classified using one Nearest Neighbor classifier. The methodology is evaluated on Flavia, ImageCLEF, ICL, Swedish leaf, and MEW2012 datasets. MEW2012 is a self-collected dataset, consisting of 153 plant species (50 leaves in each species).

Du *et al.* (2013) used an outline and venation fractal dimensions method for plant leaves recognition. The leaf edge and veins are separated by multiple thresholding based on the edge detection method. Ring projection-based wavelet fractal features, fractal dimensions of edge and venation, are extracted. K-Nearest Neighbor is applied for the classification of leaf images.

Zhang & Lei (2011) proposed a Modified Locally Linear Discriminant Embedding (MLLDE) method for recognizing the plants. It combines the advantages of Locally Linear Embedding (LLE) and Modified Maximizing Margin Criterion (MMMC) methods. Classification is performed using a 1-Nearest Neighbor method. It is observed that the MLLDE performed better compared to other feature extraction methods (Locally Linear Discriminant Embedding (LLDE) and Supervised LLE (SLLE)) in terms of accuracy.

Kadir *et al.* (2011) proposed a plant species classification methodology based on shape features (Zernike moments and Polar Fourier Transform (PFT)). Other than these features, Hu Moments and geometric features (Slimness, Roundness, and Dispersion) are also considered. Plant retrieval is based on the extracted features using Euclidean distance (between the query image and image from the database) metric. Based on the similarity, rank is assigned to each query image. The authors reported that the utilization of PFT features resulted in better performance compared to other feature combinations.

Kebapci *et al.* (2010) provided a solution for plant retrieval and identification system when there are overlapped leaves and flowers. Max Flow-Min Cut method is used for segmentation. Color features are obtained from the segmented image using color histograms and color co-occurrence matrices. Using the Scale Invariant Feature Transform (SIFT) technique, the shape features are estimated. Global shape features are extracted from the contours. The features included are the number of concave points, number of convex points, leaf base distance, leaf tip distance, leaf arc length, and normalized leaf arc length. Patch-based Gabor wavelet is used to obtain color and vein patterns. A dissimilarity score for the color feature is calculated using Kullback-Leibler (KL) divergence, number of matching SIFT key points, and L1-distance. For combined features, the dissimilarity score is calculated. This technique is

applied to a dataset of 380 plant images. 78 different plant types are included in the dataset.

Du *et al.* (2007) proposed a plant classification method known as Move Median Centers (MMC). It is a Hypersphere classifier based on digital morphological features. The digital morphological features are aspect ratio, rectangularity, the area ratio of the convex hull, perimeter ratio of convex hull, circularity, eccentricity, form factor, and invariant moments. In the MMC classifier, a median point is determined. A point closest to the median is chosen as the initial center. Then, the hyperspheres keep changing until a large radius covering all the points is determined. Redundant hyperspheres are removed. MMC classifier reduces the storage space and computational time for classification. A self-collected database of 20 species having 20 images/class is used to evaluate the method.

Du *et al.* (2006) proposed a Modified Dynamic Programming (MDP) model for Computer-Aided Plant Species Identification (CAPSI). In this approach, the Douglas-Peucker algorithm is used to obtain a smooth contour from a small number of vertices. 'Sequence of invariant attributes' representation method is chosen to represent the leaf shape. For leaf recognition, MDP is implemented. The algorithm is tested with a custom-developed database having 25 plant species. This method recognized the partial, distorted, and overlapped plant leaves.

From the literature survey, it is observed that numerous research works are reported on utilizing the geometrical features or texture features of leaf images. Most of the existing literatures use simple leaves for classification. A limited number of research works have been reported on classification for compound leaves under unconstrained environments (viewpoint or orientation variations, illumination changes, and different backgrounds).

Table 3.1 summarizes the literature survey for plant species recognition using conventional methods. It is realized that the fusion of feature extraction methods, such as Color Channel Statistics (CCS), Local Binary Pattern (LBP), Haralick Texture (HT), and Hu Moments (HM), may result in better performance for PSR system. Thus, Color (CCS), Shape (HM), and Texture Features (LBP+HT) (CSTF) algorithm are proposed for feature extraction. Later, the extracted features are classified using the Machine Learning algorithm.

Table 3.1 Summary of Literature Survey- Conventional Methods

S. No.	Feature Extraction			Classifier	Author & Year
	Shape Features	**Texture Features**	**Color Features**		
1.	MGLLDP			SVM	Wang *et al.* (2020)
2.	-	GIST Features	-	Patternnet Neural Network, SVM, K-NN	Kheirkhah & Asghari (2019)
3.	Leaf geometry and morphological features	Venation features	-	K-NN, CART, NB, Multi-SVM	Saleem *et al.* (2019)
4.	-	-	-	SVDSR	Zhang *et al.* (2018)
5.	Geometrical features, Hu moments	GLCM	-	SVM with RBF kernel	Kan *et al.* (2017)
6.	-	-	-	LSCL	Zhang *et al.* (2017)
7.	HCoS	LBPV	-	K-NN	Carranza-Rojas & Mata-Montera (2016)
8.	Directional fragment histogram	Four histograms	-	K-NN	Mzhougi *et al.* (2016)
9.	MARCH	-	-	-	Wang *et al.* (2015)
10.	I-IDSC	-	-	NN	Zhao *et al.* (2015)

Table 3.1 (Continued)

S. No.	Feature Extraction			Classifier	Author & Year
	Shape Features	**Texture Features**	**Color Features**		
11.	Geometric features, moment invariants, MDM	-	-	LDA	Kalyoncu & Toygar (2015)
12.	Aspect ratio, roundness, rectangularity, perimeter, complexity, Vein features	Contrast, Correlation, Energy, Homogeneity, entropy	Mean, Standard deviation, Skewness, Kurtosis	SVM	Ghasab *et al.* (2015)
13.	DS-LBP (texture and contour information)		-	Fuzzy-KNN	Wang *et al.* (2014)
14.	-	Leaf veins	-	RFC, SVM, PDA	Larese *et al.* (2014)
15.	Polygonal Model, B/T, HCS	-	-	-	Cerutti *et al.* (2013)
16.	Image moments, Fourier descriptors, leaf size	-	-	1-NN	Novotny & Suk (2013)
17.	Fractal dimensions, ring projection-based fractal feature	-	-	K-NN	Du *et al.* (2013)
18.	MLLDE	-	-	1-NN	Zhang & Lei (2011)
19.	Zernike moments, PFT, Hu moments, Geometric features	-	-	-	Kadir *et al.* (2011)
20.	SIFT and shape features	Patch-based Gabor Wavelet	Color histogram, color co-occurrence matrices	-	Kebapci *et al.* (2010)
21.	Morphological features and Hu moments	-	-	MMC	Du *et al.* (2007)
22.	Douglas Peucker Algorithm	-	-	MDP	Du *et al.* (2006)

3.3 METHODOLOGY

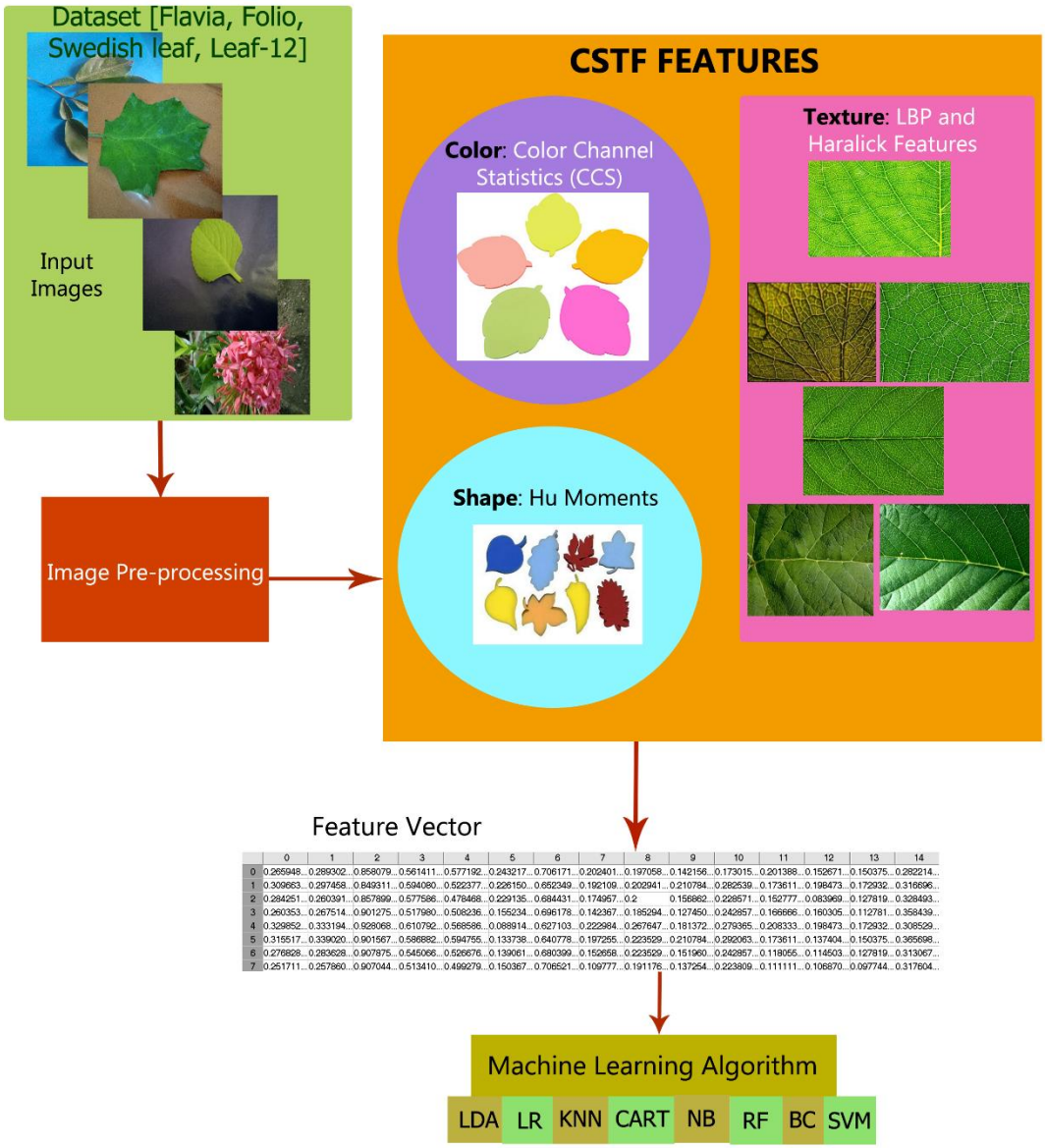


Figure 3.2 PSR System using Traditional Image Classification Method

The methodology for PSR using traditional image classification process is shown in Figure 3.2. Preprocessing is carried out on a set of input images from the dataset, and the features are extracted. Color Channel Statistics (CCS) extracts the color information (six features) from the image. Shape features are identified and recorded using Hu Moments (HM). Two methods, namely, Local Binary Pattern (LBP) and Haralick Texture (HT), are used to

extract the texture features. The Color, Shape, and Texture Features (CSTF) algorithm is the combination of the above-mentioned feature extraction methods. A single feature vector is obtained using the CSTF method. The extracted features are classified using Machine Learning classifiers such as Linear Discriminant Analysis (LDA), Logistic Regression (LR), K-Nearest Neighbor (K-NN), Classification and Regression Tree (CART), Naïve Bayes (NB), Random Forest Classifier (RFC), Bagging Classifier (BC) and Support Vector Machine (SVM).

3.3.1 Datasets

Four datasets, namely, Flavia, Folio, Swedish leaf, and Leaf-12, are utilized for evaluating the proposed method. All the images in the dataset are reconstructed to 300x300 pixels using the algorithm specified in section 2.2 (Chapter 2).

3.3.2 Pre-processing of Images

The images are resized to 100x100 pixels using a bilinear interpolation method. Bilinear interpolation is a fundamental resampling technique. This resampling method applies the distance average interpolation method. It considers four neighborhood pixels to the center point. These pixels are then averaged to produce a smoother output image.

3.3.3 Fusion of Color, Shape and Texture Features (CSTF)

The features are extracted from the resized images using the proposed CSTF method. The steps of the proposed feature extraction method are given in Algorithm 3.1. CSTF combines the features obtained from Color Channel Statistics (CCS), Hu Moments (HM), Local Binary Pattern (LBP), and Haralick Texture (HT) methods. These features are explained in detail in the following subsections. Also, the experiments are performed using the individual

feature extraction techniques and their possible combination of features. Table 3.2 shows the feature extraction method and the size of the feature vector.

Algorithm 3.1: Proposed CSTF

***Input**: Input Images, $I_0 \ldots I_N$*

***Output**: Set of Features, X; ground truth label, l*

Step 1: Read the RGB input image

Step 2: Convert RGB image to HSV image using Equations 3.1 to 3.3

Step 3: Extract the CCS features for each HSV image

Step 4: Convert the RGB image to a grayscale image

Step 5: Apply Hu Moments(HM) to extract HM_1 to HM_7 using Equations 3.4 to 3.12

Step 6: Repeat step 4. Apply Local Binary Pattern (LBP) to extract the features from the computed histogram

Step 7: Repeat step 4. Apply Haralick Textures (HT) to extract the features, HT_1 to HT_{13}

Step 8: Concatenate all the extracted features to form CSTF feature vector

$$CSTF \leftarrow concatenate\ (CCS, HM, LBP, HT)$$

Step 9: Save the features, CSTF and labels, l

Step 10: Repeat steps 1-9 for 'N' images in the dataset

Table 3.2 Summary of Feature Extraction Methods and the Size of Feature Vector

S.No.	Feature Extraction Method	Size of Feature Vector
1.	Color Channel Statistics (CCS)	6-d
2.	Hu Moments (HM)	7-d
3.	Local Binary Pattern (LBP)	36-d
4.	Haralick Texture (HT)	13-d
5.	CCS + HM	13-d
6.	CCS + LBP	42-d
7.	CCS + HT	19-d
8.	LBP + HM	43-d

Table 3.2 (Continued)

S.No.	Feature Extraction Method	Size of Feature Vector
9.	HT + HM	20-d
10.	LBP + HT	49-d
11.	CCS + LBP + HM	49-d
12.	CCS + LBP + HT	55-d
13.	CSTF	62-d

3.3.3.1 Color Channel Statistics

In Color Channel Statistics (CCS) method, the image in RGB (Red, Green, and Blue) color space is converted to HSV (Hue, Saturation, and Value) image using Equations (3.1)-(3.3) (Tang *et al.* 2017; Smith 1978). In these equations, the *MAX* and *MIN* are given by max(*R*, *G*, *B*) and min(*R*, *G*, *B*), respectively. Hue (*H*) is the color as perceived by the observer. Saturation (*S*) is the amount of intensity mixed with *H*. Value (*V*) denotes the chromatic intensity. To form the CCS feature vector, the mean (μ_H, μ_S, μ_V) and standard deviation ($\sigma_H, \sigma_S, \sigma_V$) values of *H*, *S*, and *V* channels are concatenated. Figure 3.3 shows the *H*, *S*, and *V* channel images for a sample leaf image.

$$H = \begin{cases} 0 + \frac{G-B}{MAX-MIN}.60, if\ R = MAX \\ 2 + \frac{B-R}{MAX-MIN}.60, if\ G = MAX \\ 4 + \frac{R-G}{MAX-MIN}.60, if\ B = MAX \\ undefined,\ \ if\ R = G = B \end{cases} \quad (3.1)$$

$$S = \begin{cases} 0, if\ MAX = 0 \\ \frac{MAX-MIN}{MAX}, if\ MAX \neq 0 \end{cases} \quad (3.2)$$

$$V = MAX \quad (3.3)$$

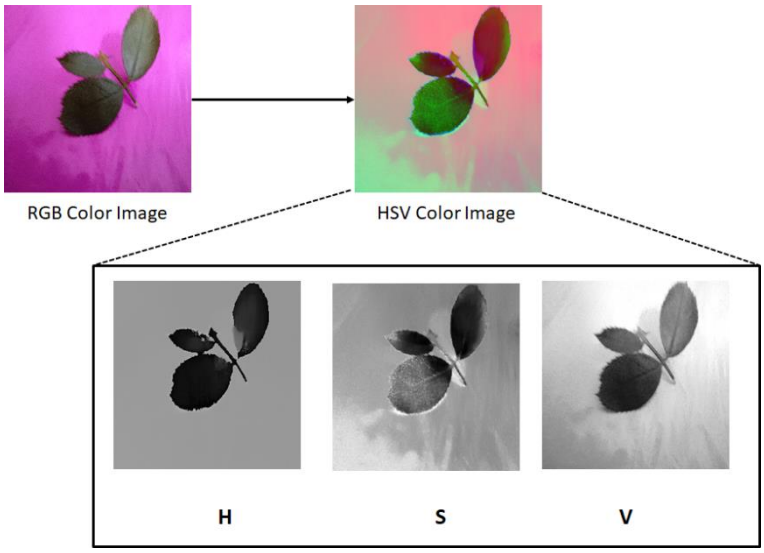


Figure 3.3 RGB image converted to HSV color space

3.3.3.2 Hu Moments

Hu Moments (HM) consists of seven Hu invariants (Hu 1962). Six moments (HM_1 to HM_6) are orthogonal invariants, and HM_7 is a skew orthogonal invariant. Equations (3.7) to (3.13) are based on normalized central moments (Equation 3.6). Normalized central moments are dependent on central moments (Equation 3.5). Geometric moment of the image *f(m, n)* is given by,

$$mo_{ij} = \sum_m \sum_n m^i n^j f(m,n) \tag{3.4}$$

If *f(m, n)* is the image and $(\bar{m}, \bar{n})$ is the centroid of the image; then the central moment is given by,

$$\psi_{ij} = \sum_m \sum_n (m-\bar{m})^i (n-\bar{n})^j f(m,n) \tag{3.5}$$

Where, $\bar{m} = \frac{mo_{10}}{mo_{00}}$; $\bar{n} = \frac{mo_{01}}{mo_{00}}$; mo_{ij} is the image geometric moment; and *i* and *j* are the integers 0,1,2…

The normalized central moment is given by,

$$\phi_{ij} = \frac{\psi_{i,j}}{\psi_{00}^{\frac{i+j}{2}+1}} \tag{3.6}$$

Hu moments are invariant to rotation, translation, and scaling (Kadir *et al.* 2011). Hu invariants are represented by the following equations,

$$HM_1 = \phi_{20} + \phi_{02} \quad (3.7)$$

$$HM_2 = (\phi_{20} + \phi_{02})^2 + 4\phi_{11}{}^2 \quad (3.8)$$

$$HM_3 = (\phi_{30} - 3\phi_{12})^2 + (3\phi_{21} - \phi_{03})^2 \quad (3.9)$$

$$HM_4 = (\phi_{30} + \phi_{12})^2 + (\phi_{21} + \phi_{03})^2 \quad (3.10)$$

$$HM_5 = (\phi_{30} - 3\phi_{12})(\phi_{30} + \phi_{12})[(\phi_{30} + \phi_{12})^2 - 3(\phi_{21} + \phi_{03})^2] + (\phi\mu_{21} - \phi_{03})(\phi_{21} + \phi_{03})[3(\phi_{30} + \phi_{12})^2 + (\phi_{21} + \phi_{03})^2] \quad (3.11)$$

$$HM_6 = (\phi_{20} + \phi_{02})[(\phi_{30} + \phi_{12})^2 - (\phi_{21} + \phi_{03})^2] + 4\phi_{11}(\phi_{30} + \phi_{12})(\phi_{21} + \phi_{03}) \quad (3.12)$$

$$HM_7 = (3\phi_{21} - \phi_{03})(\phi_{30} + \phi_{12})[(\phi_{30} + \phi_{12})^2 - 3(\phi_{21} + \phi_{03})^2] \quad (3.13)$$

3.3.3.3 Local Binary Pattern

Local Binary Pattern (LBP) (Ojala *et al.* 2000) extracts the texture content from grayscale images. The neighborhood is chosen around the center pixel in the grayscale image. Traditionally, the image pixels are thresholded on comparing the center pixel with the neighboring pixel, for every 3x3 window. It results in a binary number pattern.

An extension of LBP is proposed to handle the variation in neighborhood size. For a circular symmetric neighborhood, radius r and number of points *pt* around the center is considered. The diagonal gray pixels are determined using an interpolation method. Unique rotationally invariant patterns termed as prototypes are obtained.

In this chapter, the results of LBP are reported for *r*=6, *pt*=8. It results in 36 unique patterns. Figure 3.4 (Source: Ojala *et al.* 2000) shows the

36 unique patterns, where black dots represent ‘0’ bit, and white circles represent ‘1’ bit. Each unique pattern forms a binary number. Thus, 36 bins of histogram features are obtained. Figure 3.5 shows the output of a rotational invariant LBP image.

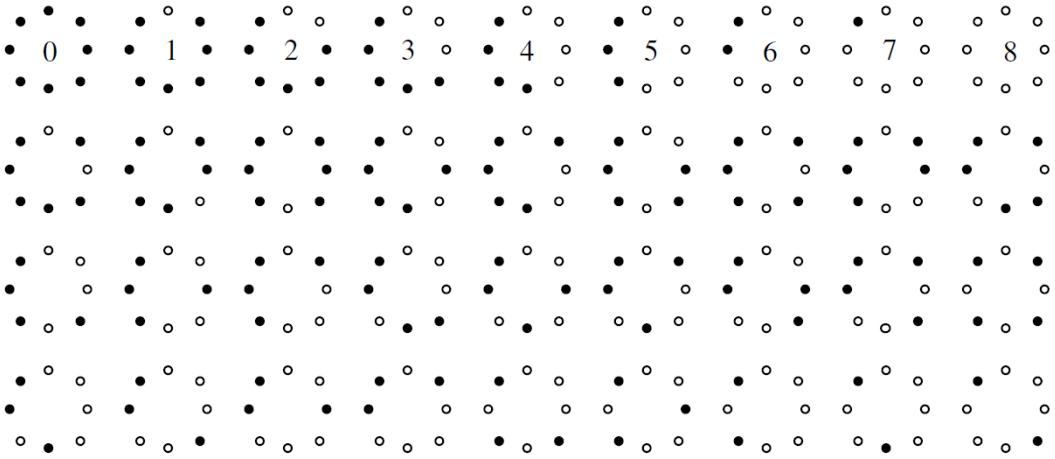


Figure 3.4 36 Unique patterns for rotational invariant LBP (Source: Ojala *et al.* 2000)

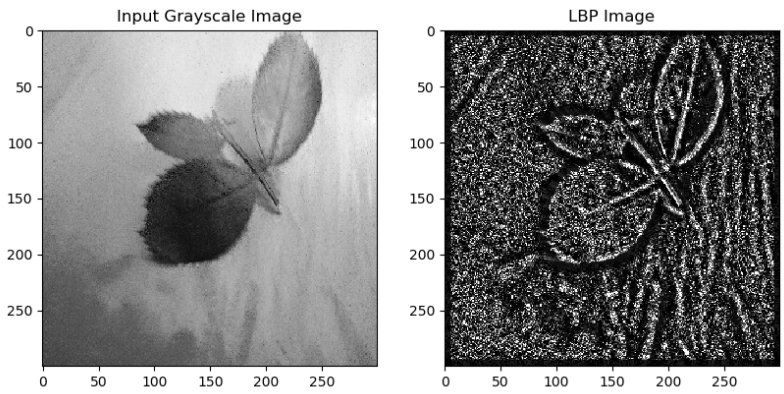


Figure 3.5 Output of LBP for Grayscale Input Image

3.3.3.4 Haralick Texture

Haralick Texture (HT) is an image descriptor based on the Gray-Level Co-Occurrence Matrix (GLCM) introduced by Haralick *et al.* (1973). HT is rotationally invariant. The matrix is used to determine the number of two adjacent gray-level pixels in an image. The adjacency of gray-pixels is tested in

four directions (left to right, right to left, top-left to bottom-right, top-right to bottom-left) (Mohanraj *et al.* 2018). Each direction results in a GLCM matrix. From the GLCM matrices, the Haralick features (HT_1-HT_{13}) are calculated. It is used as a 13-d feature vector. The thirteen HT features are tabulated in Table 3.3. Due to the instability of the maximal correlation coefficient (HT_{14}), this feature is not considered.

In Table 3.3, $p(i,j)$ represents the (i,j)th entry of normalized gray spatial-dependent matrix, ($p(i,j)/nc$), where nc is a normalizing constant. Gr is the number of gray levels from the quantized image. $p_m(i)$ is the i^{th} entry of the marginal probability matrix, on summing all rows from $p(i,j)$, $p_m(i) = \sum_{j=1}^{Gr} p(i,j)$. $p_n(j)$ is j^{th} entry of marginal probability (i,j) matrix, on summing all columns from $p(i,j)$ matrix, $p_n(j) = \sum_{i=1}^{Gr} p(i,j)$. Also, $p_{(m+n)}(e) = \sum_{i=1}^{Gr}\sum_{j=1}^{Gr} p(i,j)\,, e = 2,3, \ldots 2Gr$; $i+j=e$ and $p_{(m-n)}(e) = \sum_{i=1}^{Gr}\sum_{j=1}^{Gr} p(i,j)\,, e = 0,1 \ldots Gr-1$; $|i-j|=e$.

Table 3.3 Haralick Texture features

Feature	Statistical Feature	Formula
HT_1	Angular Second Moment (ASM)	$HT_1 = \sum_i \sum_j \{p(i,j)\}^2$
HT_2	Contrast	$HT_2 = \sum_{o=0}^{Gr-1} o^2 \left\{ \sum_{i=1}^{Gr} \sum_{\substack{j=1 \\ \lvert i-j \rvert = o}}^{Gr} p(i,j) \right\}$,
HT_3	Correlation	$HT_3 = \frac{\sum_i \sum_j (ij)p(i,j) - \mu_m \mu_n}{\sigma_m \sigma_n}$ μ_m, μ_n are means of p_m and p_n σ_m, σ_n are standard deviations of p_m and p_n
HT_4	Sum of Squares	$HT_4 = \sum_i \sum_j (i-\mu)^2 p(i,j)$
HT_5	Inverse Difference Moment	$HT_5 = \sum_i \sum_j \frac{1}{1+(i-j)^2} p(i,j)$

Table 3.3 (Continued)

Feature	Statistical Feature	Formula
HT_6	Sum Average	$HT_6 = \sum_{i=2}^{2Gr} ip_{m+n}(i)$
HT_7	Sum Variance	$HT_7 = \sum_{i=2}^{2Gr} (i - HT_8)^2 p_{m+n}(i)$
HT_8	Sum Entropy	$HT_8 = -\sum_{i=2}^{2Gr} p_{m+n}(i) log\{p_{m+n}(i)\}$
HT_9	Entropy	$HT_9 = -\sum_i \sum_j p(i,j) \log(p(i,j))$
HT_{10}	Difference Variance	$HT_{10} = Var(p_{m-n})$
HT_{11}	Difference Entropy	$HT_{11} = -\sum_{i=0}^{Gr-1} p_{m-n}(i) log\{p_{m-n}(i)\}$
HT_{12}	Information Measure of Correlation	$HT_{12} = \frac{HXY - HXY1}{max\{HX, HY\}}$ Here, $HXY = -\sum_i \sum_j p(i,j) \log(p(i,j))$
HT_{13}	Information Measure of Correlation	$HT_{13} = (1 - \exp[-2.0(HXY2 - HXY)])^{1/2}$

3.3.4 Classification Methods

Several machine learning classifiers are applied to the extracted features. The classifiers used in the analysis are Linear Discriminant Analysis (LDA), Multinomial Logistic Regression (LR), K-Nearest Neighbor (K-NN), Classification and Regression Tree (CART), Naïve Bayes (NB), Random Forest Classifier (RFC), Bagging Classifier (BC) and Support Vector Machine (SVM). The classifiers are discussed in detail (subsections 3.3.4.1-3.3.4.7).

3.3.4.1 Linear Discriminant Analysis (LDA)

LDA is a statistical method. It is used for classification and feature reduction. It supports the multi-class classification. LDA computes the mean

and covariance values for the entire dataset. It is used to identify the scatter within a class and between the classes.

Bayes theorem (Marsland 2014), as stated in Equation (3.13), is applied to estimate the probabilities in LDA (Pedregosa *et al.* 2005). For the experimental analysis, 'SVD' (Singular Value Decomposition) is used as a solver (Pedregosa *et al.* 2005).

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)} \qquad (3.14)$$

Where, *P (A|B)* is the conditional probability (Gose 1997) for the occurrence A given B,

P (B|A) is the conditional probability for the occurrence *B* given *A*,

P (A) and *P (B)* are the probabilities of *A* and *B*

3.3.4.2 Multinomial Logistic Regression (LR)

LR predicts the outcome for binary classification problems. Whereas, multinomial LR (Krishnapuram *et al.* 2005) is used for classifying multiple classes. The experimental results are recorded for multinomial LR. Multinomial LR treats the multi-class problem as multiple binary LR models. Probabilities obtained from the binary LR models are combined to form the result of Multinomial LR. Equation (3.15) (Böhning 1992; Krishnapuram *et al.* 2005) represents the probability of sample data '*x*' belonging to the class *C* ($c\epsilon\{1,2 \dots, C\}$ classes), y_i is the class labels, and w_i is the weight vector of class *C*.

$$P(y_i = 1|x, w) = \frac{\exp(w_{(i)}^T x)}{\sum_{c=1}^{C} \exp(w_{(c)}^T x)} \qquad (3.15)$$

3.3.4.3 Naïve Bayes (NB)

Naïve Bayes (NB) (Marsland 2014) is a supervised learning algorithm, which is simple and straightforward. It supports the multi-class classification. NB is termed with 'naïve' as one feature, and it is independent of other features. The probability of each feature contributes to the classification. NB is based on Bayes theorem. Bayes theorem is stated in Equation 3.16.

Let us consider the features x_1, $x_2 \ldots x_f$ belonging to Class, C, where there are $c \epsilon \{1,2 \ldots, C\}$ classes. Classification is performed using the estimated conditional probabilities. Equation (3.13) (Marsland 2014) is rewritten for n number of features in c_i as follows,

$$P(c_i|x_1, x_2 \ldots x_f) = \frac{P(x_1, x_2 .. x_f|c_i)}{P(x_1, x_2 \ldots x_f)} . P(c_i) \quad (3.16)$$

Equation (3.16) is simply rewritten as,

$$P(c_i|x_1, x_2 .. x_f) \propto \left(\prod_{j=1}^{j=f} P(x_f|c_i)\right) . P(c_i), \forall\ 1 \leq c \leq C \quad (3.17)$$

Gaussian Naïve Bayes algorithm is used in the experimental investigation. This type of NB is presumed that the features follow a Normal Distribution (Pedregosa *et al.* 2005).

3.3.4.4 K-Nearest Neighbor (K-NN)

K-Nearest Neighbor (K-NN) (Mitchell 1997) is a supervised learning algorithm. The class label (y_i) of the query image (q_i) is determined based on its adjacent neighbor with high similarity. K-NN utilizes the Euclidean distance metric, and it is given by,

$$d(ne_i,\ q_i) = \sqrt[2]{(ne_i - q_i)^2} \quad (3.18)$$

Where $d(ne_i, q_i)$ is the distance between the query image, q_i, and the neighboring image, ne_i. The distances between ne_i and q_i are sorted in ascending

order. If k=1, assign y of the single nearest neighbor. Else if k>1, majority voting rule is applied to the nearest neighbors to determine their class label, y. 'k' is a hyperparameter. The performance analysis of the PSR system is carried out for different values of k to identify the best k (between 1 to 50). The k value with the largest performance metric is chosen to be the final k value of the model. The model is retrained with the best k value before it is explored to the testing data.

3.3.4.5 Classification and Regression Tree (CART)

Classification and Regression Tree (CART) (Marsland 2014) is coined by Leo Breiman. It is a supervised predictive machine-learning algorithm. CART builds the binary trees, as it is computationally cost-effective (O (log N), N representing the number of data points). CART utilizes the if-then logical conditions to predict the class.

In CART, every internal node performs a test on the attributes. The link between the nodes represent decisions, and the leaf node holds the class label. This algorithm selects and assigns the features required for classification as a root node. A decision is made at the root node and internal nodes. Gini impurity function is calculated to determine the purity of the leaves. Entropy is used for information gain.

$$Gini = 1 - \sum_{c=1}^{C} P_c^2 \quad (3.19)$$

$$Entropy, H(E) = -\sum_{c=1}^{C} P_c log P_c \quad (3.20)$$

In Equations (3.19) and (3.20), c varies from 1 to C number of classes, and P_c is the class probability.

3.3.4.6 Random Forest Classifier (RFC)

Random Forest Classifier (RFC) (Breiman 2001; Marsland 2014) is a supervised learning algorithm. It applies to both classification and regression

problems. RFC classifier is a forest of CARTs or Decision Trees. Using CART, randomly selected data samples are utilized in prediction. The best solution is obtained by the 'voting scheme'. For all the experiments using RFC, the number of trees is set as 200. Algorithm 3.2 shows the steps in RFC. This classifier does not overfit the model, and it does not require a k-fold cross-validation function since the classifier considers only random samples. RFC handles huge amount of data, extracts feature importance information. It also reduces overall variance and error. RFC has little impact on outliers. For very large datasets, the size of the trees utilize a lot of memory.

Algorithm 3.2: RFC Classification Algorithm

Input: Features, $x_0 \ldots . x_f$

Output: Predicted class label, y_i

Step 1: Select random samples from a given dataset.

Step 2: Construct multiple decision trees for each sample and get a prediction result from each of the decision trees.

for each tree:

i) At each node of the decision tree, randomly select x_f features, and compute the Information gain (or Gini impurity) only on that set of features, selecting the optimal one.

ii) Repeat until the tree is complete

Step 3: Perform a vote for each predicted result.

Step 4: Select the prediction result with the most number of votes as the final prediction and assign the class label, y_i.

3.3.4.7 Bagging Classifier (BC)

'Bootstrap Aggregating' is called as a Bagging Classifier (BC) (Marsland 2014). It constructs *M* number of individual models. BC uses *M* number of CARTs. The training data is resampled multiple times with replacement strategy. The major variation between RFC and BC is that all

features are considered for splitting a node in BC. While in RFC, random subset of features are chosen and best feature is used for splitting the node. The prediction results of each CART are combined using averaging process to form the final prediction. BC reduces the variance obtained in the individual CARTs. BC does not suit underfitting data. Also, BC ignores highest and lowest result for averaging process.

3.3.4.8 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm introduced by Vapnik in the year 1995. The main idea of SVM (Jothi & Bhargavi 2019) is to use a maximum marginal hyperplane to separate the dataset into distinct classes.

The hyperplane is the optimal decision boundary that differentiates the support vectors (data points) of each class. The kernel functions are used to find the hyperplane. For SVM classifier, the linear kernel is used with c=1 and one-vs-one ('ovo') (Pedregosa *et al.* 2005; Galar M *et al.* 2013) decision scheme for multi-class classification. The linear kernel is given by,

$$K(w, x) = 1 + w^T x \tag{3.21}$$

3.4 RESULTS AND DISCUSSION

Four datasets, namely, Flavia (Wu *et al.* 2007), Folio, Swedish leaf (Soderkvist 2002), and Leaf-12 datasets, are used in the performance analysis of the PSR system using traditional image classification methods. The train and test split ratio is 70:30. The implementation is carried out using the i7 processor with 20 GB RAM and NVIDIA Titan X GPU. Python, OpenCV, and supporting packages (Scikit-learn (Pedregosa *et al.* 2005), Mahotas, Numpy, H5py, OS, Matplotlib, and Seaborn) are used.

3.4.1 Evaluation Metrics

Performance metrics such as Accuracy, Precision, Recall, F1-Score, and Confusion Matrix are considered.

3.4.1.1 Classification Accuracy

Classification accuracy of a model is defined based on the number of images correctly predicted to that of the total number of test images, as specified in Equation (3.22) (Wäldchen & Mäder 2018).

$$Accuracy = \frac{Number\ of\ Images\ Correctly\ Predicted}{Total\ number\ of\ Test\ Images} \times 100 \quad (3.22)$$

(or)

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.23)$$

Where *TP* is True Positive, *TN* is True Negative, *FP* is False Positive, and *FN* is False Negative.

The performance analysis of the PSR system is based on the obtained top-1 and top-5 accuracies. Top-1 accuracy is obtained by comparing the predicted label with the ground truth value. The total number of matched labels (predicted, ground truth) is used in the estimation of *top-1* accuracy. In *Top-5* accuracy, the target label is one among the top five (*n*=5) predictions.

Average *Top-n* accuracy (Barre *et al.* 2017) is given by,

$$Top - n = \frac{1}{N}.\sum_{i=1}^{N}\sum_{j=1}^{n} T(f(I_i, w) = y_i) \quad (3.24)$$

Where, *N* is the number of samples for testing; *n*=1, 5 or 10;

y_i is the predicted class for the test images, I_i, *i*=1, 2,.., *N* and *j*=1,2,…*n*. *T(.)* is an indicator function returning either 0 or 1.

3.4.1.2 Precision

Precision, *Pr* (Marsland 2014) is the ratio of *TP* to the sum of *TP* and *FP*. It is represented in Equation (3.25). When the precision is high, the classifier's performance is considered to be more relevant.

$$Pr = \frac{TP}{TP+FP} \tag{3.25}$$

3.4.1.3 Recall

Recall, *Re* (Marsland 2014) is the number of *TP*s divided by the sum of *TP*s and *FN*s. Higher recall value denotes that the classifier's performance obtained on the images are considered to be more relevant.

$$Re = \frac{TP}{TP+FN} \tag{3.26}$$

3.4.1.4 F1-Score

F1-score, *F1* (Marsland 2014) is the harmonic mean of precision and recall. This value is given in Equation (3.27).

$$F1 = 2.\frac{Pr*Re}{Pr+Re} \tag{3.26}$$

3.4.1.5 Confusion Matrix

A confusion matrix (Marsland 2014) is a table that describes the predictions of the model on test images. It is the simplest way to measure the performance of a classifier. For binary classification, the confusion matrix is represented as in Figure 3.6. *TP* is the number of test images correctly predicted as a positive class. *TN* denotes the number of test images correctly predicted as negative class. The mispredicted positive class is represented as *FP*. Similarly, the mispredicted negative class is *FN*.

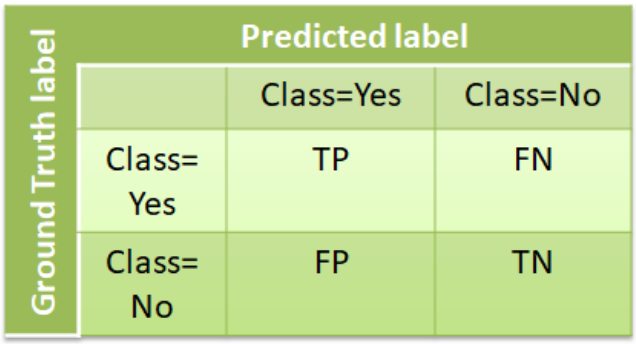


Figure 3.6 Sample Confusion Matrix

3.4.2 Flavia dataset

Table 3.4 Results of Flavia Dataset

Feature Extractor	Machine Learning Algorithm							
	LDA	**LR**	**NB**	**K-NN**	**CART**	**RFC**	**BC**	**SVM**
Color Channel Statistics (CCS)	49.38	31.25	47.29	51.67	47.29	**57.29**	56.88	45.42
Hu Moments (HM)	35.21	16.25	33.33	44.38	42.50	**56.25**	53.33	31.04
Local Binary Pattern (LBP)	**66.88**	58.75	53.75	56.04	38.75	57.92	50.42	62.92
Haralick Textures (HT)	**73.54**	40.83	56.67	61.25	53.75	64.79	60.00	51.25
CCS+HM	66.67	47.5	56.25	67.71	59.17	**77.71**	70.62	57.50
CCS+LBP	**78.54**	72.08	63.75	69.38	54.17	74.58	67.71	75.62
CCS+HT	**80.21**	55.21	59.79	71.46	65.00	75.21	71.25	65.42
LBP+HM	**76.67**	66.04	62.92	62.92	52.50	74.79	67.08	70.21
HT+HM	**79.58**	56.88	60.00	72.08	63.75	78.54	73.75	62.50
LBP+HT	**83.75**	70.00	65.00	63.75	55.00	74.58	66.67	74.17
CCS+LBP+HM	**82.29**	75.42	67.29	71.46	59.58	82.08	72.29	79.17
CCS+LBP+HT	**86.88**	76.04	68.33	72.71	62.50	81.04	78.75	80.42
CSTF (Proposed Method- CCS+ LBP + HT+ HM)	**89.17**	78.96	70.21	75.83	65.62	85.62	77.50	83.75

Table 3.4 shows the results obtained for Flavia dataset. For CCS (57.29%) and HM (56.25%) features, RFC shows better accuracies compared to

other classifiers. Whereas, for both the individual texture features (LBP-66.88%, HT-73.54%), the LDA classifier performed better. It is also higher than the CCS and HM based classification methods.

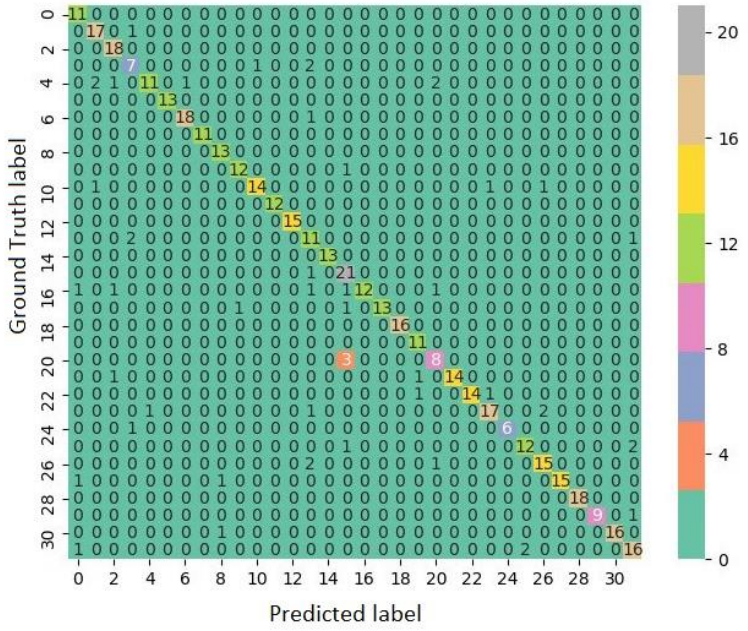


Figure 3.7 Confusion Matrix obtained by using CSTF method + LDA classifier

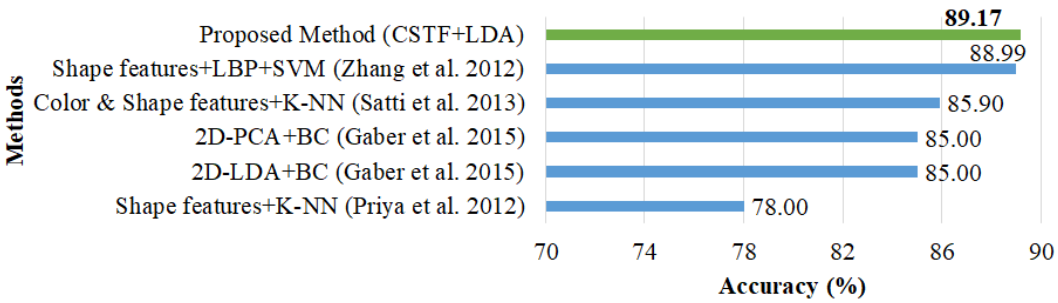


Figure 3.8 Comparison of Existing Methods with CSTF + LDA classifier

For a combination of two feature extraction methods, it is observed that the LBP+HT and CCS+HT resulted in accuracies above 80% for the LDA classifier. Except for CCS+HM (RFC-77.71%), all other two combination

features produced higher accuracies by utilizing the LDA classifier. It is observed that the texture features contribute more to the classification process.

Among the combinations CCS+LBP+HM and CCS+LBP+HT, the latter with LDA resulted in an accuracy of 86.88%. The fusion of Color Channel Statistics (CCS), Hu Moments (HM), Haralick Texture (HT), and Local Binary Pattern (LBP) resulted in higher accuracy compared to other combinations of features. The CSTF with LDA classifier attained an accuracy of 89.17%.

As far as the classifiers are concerned, LDA outperforms other classification methods. Figure 3.7 depicts the confusion matrix obtained by using the CSTF method + LDA classifier.

Figure 3.8 shows the comparison of CSTF+LDA method with other existing methods, utilizing the Flavia dataset. Priya *et al.* (2012) reported an accuracy of 78% by utilizing the shape features with K-NN. Gaber *et al.* (2015) reported accuracies of two feature extraction methods (2D-LDA, 2D-PCA) with BC as 85%.

An accuracy of 88.99% is obtained for SVM classifier using shape-based features and LBP (Zhang *et al.* 2012). It differs from the CSTF + LDA method by 0.18%. Satti *et al.* (2013) attained an accuracy of 85.90% for Color and Shape features with K-NN. To summarize, the proposed method (CSTF+LDA) outperforms other existing approaches for the Flavia dataset.

3.4.3 Folio dataset

The experimental results for the Folio dataset are listed in Table 3.5. For the individual features considered, CCS + RFC and HM + RFC resulted in an accuracy of 54.91% and 42.2%, respectively. The texture feature HT + LDA (63.58%), resulted in higher accuracies compared to CCS, HM, and LBP methods.

Among the double feature combinations, the feature extraction methods (CCS + HM, CCS + LBP, HT + HM, and LBP + HM) with Random Forest Classifier resulted in higher accuracy compared to other classifiers. It is also observed that the combination of features (CCS + HT and LBP + HT) with the LDA classifier resulted in higher accuracy compared to other classifiers.

Table 3.5 Results of Folio Dataset

Feature Extractor	Machine Learning Algorithm							
	LDA	**LR**	**NB**	**K-NN**	**CART**	**RFC**	**BC**	**SVM**
Color Channel Statistics (CCS)	51.45	32.37	51.45	53.18	45.66	**54.91**	53.76	35.84
Hu Moments (HM)	29.48	13.29	34.10	34.68	35.84	**42.20**	36.42	21.97
Local Binary Pattern (LBP)	**52.02**	38.15	38.15	38.73	30.06	48.55	42.77	43.35
Haralick Textures (HT)	**63.58**	35.26	56.07	52.02	49.13	60.69	59.54	40.46
CCS+HM	60.12	34.10	57.80	60.12	55.49	**71.10**	61.85	37.57
CCS+LBP	62.43	51.45	53.18	50.29	46.24	**72.25**	63.01	53.76
CCS+HT	**73.99**	46.24	63.01	67.63	57.80	65.90	64.16	54.91
LBP+HM	57.80	41.62	46.82	39.31	36.42	**61.85**	49.13	45.09
HT+HM	68.79	35.84	57.23	61.85	51.45	**69.94**	63.58	45.66
LBP+HT	**72.83**	48.55	51.45	45.66	47.40	68.79	57.23	55.49
CCS+LBP+HM	68.79	52.60	59.54	56.07	54.34	**75.14**	65.32	54.91
CCS+LBP+HT	**76.30**	60.12	58.96	59.54	56.65	72.25	67.05	64.74
CSTF (Proposed Method- CCS+ LBP+ HT+ HM)	**79.77**	60.12	60.12	60.69	45.66	78.03	71.68	64.74

The accuracy of CCS+LBP+HT with the LDA classifier is about 76.3%. It is greater than CCS+LBP+HM with RFC by 1.16%. For Folio dataset, CSTF features (CCS+LBP+HT+HM) along with LDA classifier outperformed all the other combinations with an accuracy of 79.77%. The confusion matrix obtained by using the CSTF + LDA classifier is displayed in Figure 3.9.

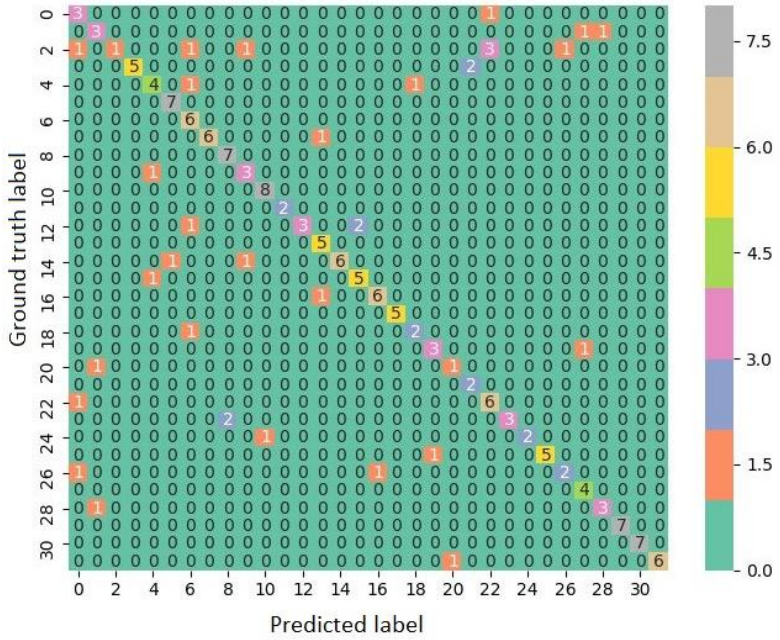


Figure 3.9 Confusion Matrix obtained by using CSTF + LDA classifier

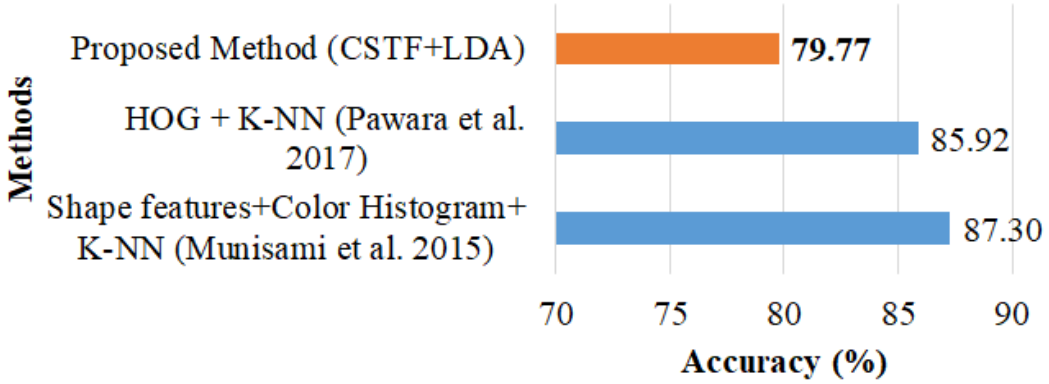


Figure 3.10 Comparison of Existing Methods with CSTF + LDA

From Figure 3.9, it is observed that the number of mispredictions is high. This might be due to the less number of input images per class. The misprediction by the model could be reduced by increasing the number of input images in the dataset.

The proposed method (CSTF+LDA) showed an accuracy of 79.77% for the Folio dataset. The authors of the Folio dataset, Munisami *et al.* (2015), reported an accuracy of 87.3% for shape features, color histogram along with

K-NN. Pawara *et al.* (2017) obtained an accuracy of 84.30±1.62 for Histogram of Gradients (HoG) with K-NN. The classification accuracies obtained by Munisami *et al.* (2015) and Pawara *et al.* (2017) are higher than CSTF features with LDA classifier.

3.4.4 Swedish leaf dataset

Table 3.6 Results of Swedish Leaf Dataset

Feature Extractor	Machine Learning Algorithm							
	LDA	**LR**	**NB**	**K-NN**	**CART**	**RFC**	**BC**	**SVM**
Color Channel Statistics (CCS)	**68.34**	47.34	54.44	51.18	50.00	60.06	57.40	52.66
Hu Moments (HM)	58.88	36.69	50.59	63.31	52.96	**66.86**	61.83	52.07
Local Binary Pattern (LBP)	**75.15**	72.78	61.24	61.24	50.00	70.41	64.50	72.78
Haralick Textures (HT)	**86.69**	63.31	61.83	73.67	62.72	76.04	72.78	66.86
CCS+HM	**83.14**	62.43	60.95	72.49	66.27	77.51	74.56	69.82
CCS+LBP	**85.80**	79.29	68.05	73.08	63.91	77.81	73.37	80.77
CCS+HT	**88.76**	71.30	71.60	77.22	75.15	82.54	79.29	77.22
LBP+HM	**83.14**	77.22	66.27	67.46	67.16	79.88	73.08	81.07
HT+HM	**92.01**	74.56	67.46	82.84	76.63	83.14	79.59	77.81
LBP+HT	**83.73**	79.59	68.05	73.37	65.38	81.07	77.22	82.84
CCS+LBP+HM	**90.53**	83.73	70.41	77.81	70.41	83.73	79.29	84.62
CCS+LBP+ HT	**86.39**	83.73	72.49	79.29	68.64	86.09	79.29	84.32
CSTF (Proposed Method- CCS+ LBP+ HT+ HM)	**88.46**	85.80	73.96	81.07	80.18	87.87	85.50	88.17

Table 3.6 summarizes the results obtained for the Swedish leaf dataset. The individual feature extraction methods (CCS, HT, and LBP) along with LDA classifier resulted in accuracies of 68.34%, 86.69%, and 75.15%, respectively. The texture feature method (HT) with LDA classifier obtained higher accuracy compared to other individual feature extraction methods. The accuracy of HM with RFC is 66.86%.

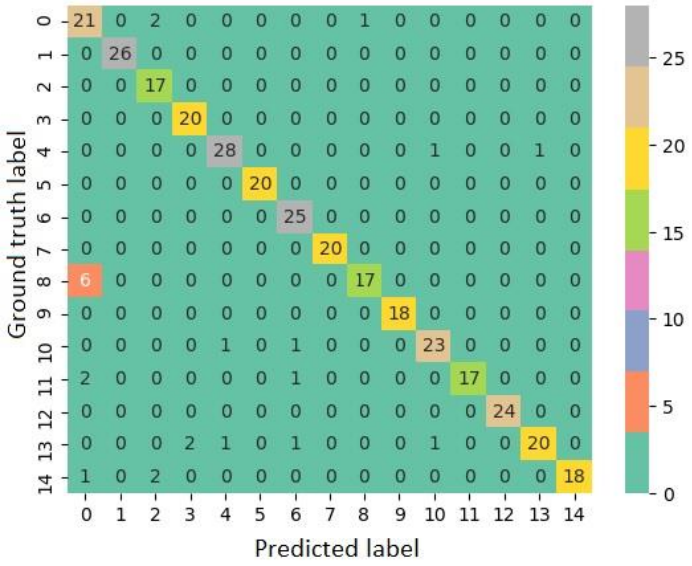


Figure 3.11 Confusion Matrix obtained by using HT + HM + LDA

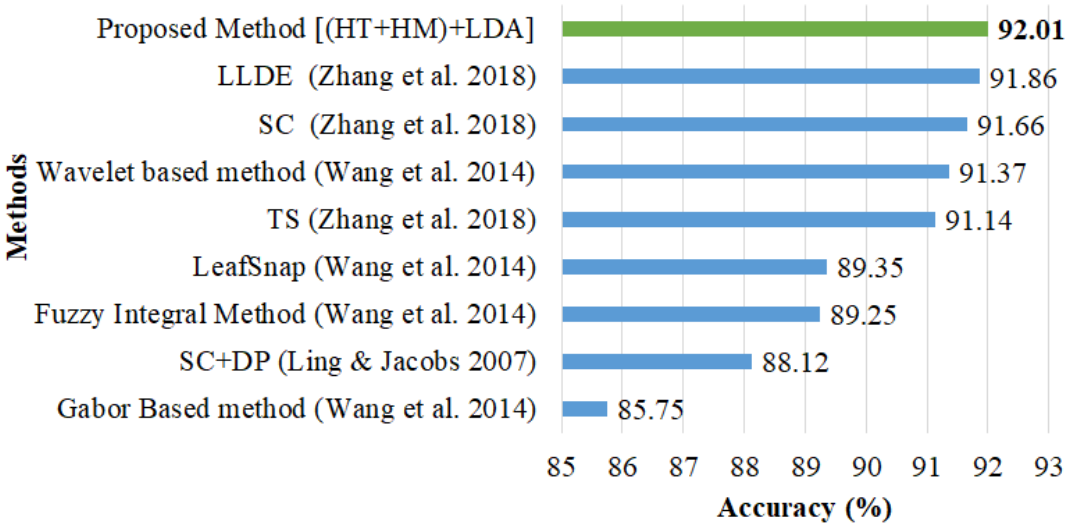


Figure 3.12 Comparison of (HT+HM) + LDA with Existing literature

Among the double feature combinations, the highest accuracy (92.01%) is obtained for HT+HM with LDA classifier. The accuracy of CCS+HT with LDA is 88.76%. The double-feature combinations (CCS+HM and LBP+HM) with LDA classifier resulted in identical accuracy of 83.14%. The texture feature combination (LBP+HT) with LDA classifier obtained an accuracy of 83.73%.

In triple feature combinations, the performance of CCS+LBP+HM with LDA classifier (90.53%) surpasses the CCS+LBP+HT with LDA classifier (86.39%). The purpose of the feature extraction method (CSTF- CCS + HM + HT + LBP) with LDA classifier does not improve the performance of the PSR system. From Table 3.6, it is observed that the shape (HM) and texture feature (HT) with LDA classifier, outperforms all the other combinations.

Figure 3.11 shows the confusion matrix obtained by using HT+HM features with LDA classifier. In Figure 3.11, it is observed that there is more number of mispredictions for class 8 (class 8 is mispredicted as class 0). *Top-5* accuracy obtained by using HT + HM features with LDA classifier is 99.7%.

Figure 3.12 indicates the comparison of existing approaches with the result obtained by using HT + HM features with LDA classifier. SC (Shape Context) + DP (Dynamic Programming) (Ling & Jacobs 2007) obtained an accuracy of 88.12%. Wang *et al.* (2014) reproduced certain existing methods such as Gabor based method, Fuzzy Integral method, LeafSnap, and Wavelet-based method using the Swedish leaf dataset. (HT+HM)+LDA outperforms these four methods in terms of accuracy.

Accuracy of Texture and Shape Features (TS) (Zhang *et al.* 2018) approach is lesser than (HT+HM) + LDA by 0.87%. Local Linear Discriminant Embedding (LLDE-90.28±1.58%) (Zhang *et al.* 2018), Shape and Color Features (90.32±1.34) (Zhang *et al.* 2018) and Wavelet-based method, reported an accuracy closer to the value obtained by using the HT+HM with LDA classifier.

3.4.5 Leaf-12 dataset

The experimental results for the Leaf-12 dataset are listed in Table 3.7. For individual features, CCS with RFC resulted in higher accuracy of 81.99%. Among the two feature combinations, the highest accuracy of 82.38%

is obtained for CCS+HT with RFC. For double feature combination, CCS+HM with RFC classifier produced an accuracy of 82.11%, and it varies by 0.27% compared to CCS+HT with RFC classifier.

Table 3.7 Results of Leaf-12 Dataset

Feature Extractor	Machine Learning Algorithm							
	LDA	**LR**	**NB**	**K-NN**	**CART**	**RFC**	**BC**	**SVM**
Color Channel Statistics (CCS)	32.38	32.81	35.50	80.08	68.14	**81.99**	73.18	36.55
Hu Moments (HM)	13.80	10.68	12.76	27.43	35.42	**48.44**	40.89	11.46
Local Binary Pattern (LBP)	41.41	41.23	28.82	45.23	30.21	**51.74**	43.75	42.27
Haralick Textures (HT)	33.68	23.61	22.14	57.73	43.92	**57.64**	53.91	25.17
CCS+HM	40.10	35.68	17.27	81.86	65.62	**82.11**	74.57	37.33
CCS+LBP	51.22	52.08	36.72	73.78	57.64	**78.12**	68.84	56.34
CCS+HT	47.31	45.57	34.98	79.08	67.19	**82.38**	75.69	49.91
LBP+HM	44.53	42.53	21.96	47.74	35.76	**64.58**	52.43	42.71
HT + HM	36.11	29.08	18.06	63.98	51.74	**69.97**	63.19	29.77
LBP + HT	53.91	47.83	35.24	62.85	44.88	**69.53**	58.77	50.95
CCS + LBP + HM	54.95	53.56	28.30	73.70	59.72	**79.95**	71.09	56.68
CCS + LBP + HT	62.24	57.9	40.54	77.43	59.90	**81.42**	73.35	63.72
CSTF (Proposed Method- CCS +LBP+HT+HM)	63.72	58.77	31.86	77.86	62.50	**82.38**	74.22	64.32

CCS+LBP+HT with RFC resulted in an accuracy of 81.42% and is higher than CCS+LBP+HM with RFC, with a variation of 1.47%. Random Forest classifier resulted in higher accuracy compared to all the other classifiers, irrespective of the combination of features and techniques.

The features using Color, Shape, and Texture Fusion (CSTF) with Random Forest classifier resulted in higher accuracy (82.38%) compared to all the other methods. The complexity (more number of images, variation in orientation, lighting conditions, and different backgrounds) of the Leaf-12 dataset is high in comparison with the other three standard datasets (Flavia, Folio, and Swedish leaf datasets). In this perspective, the performance of the

Leaf-12 dataset for traditional classification is lower compared to other datasets.

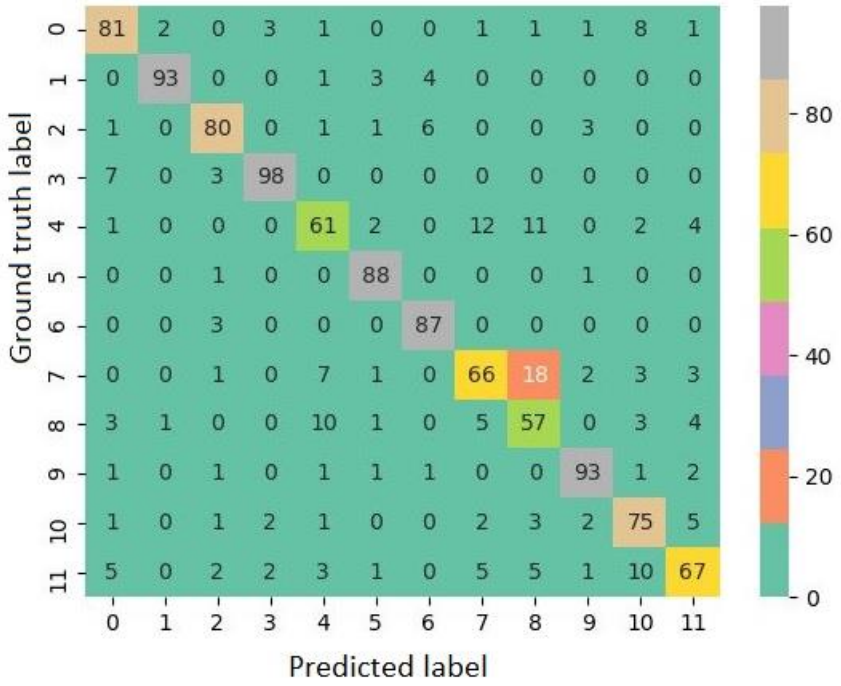


Figure 3.13 Confusion Matrix by using CSTF + RFC

Figure 3.13 indicates the confusion matrix obtained by using the CSTF + RFC. The number of mispredictions for different classes is high. *Top-1* accuracy of CSTF with RFC is 82.38%.

3.5 SUMMARY

In this chapter, the performance analysis of the PSR system using different feature extraction methods with various Machine Learning classifiers is reported. The resizing of images in the dataset is performed using a bilinear interpolation method. To the resized images, the features (one feature- CCS, LBP, HM, and HT; two features-CCS + HM, CCS + LBP, CCS + HT, LBP + HM, HT + HM and LBP+HT; three features- CCS + LBP + HM and CCS + LBP + HT; CSTF-CCS + LBP + HT + HM) are extracted. Machine Learning algorithms are applied to the extracted features.

The best result for each dataset is summarized in Table 3.8. For the Flavia dataset, CSTF features with LDA classifier achieved better performance metrics. *Top-1* accuracy, *top-5* accuracy, precision (*Pr*), recall (*Re*), f1-score (*F1*) obtained are 89.17%, 98.96%, 0.91, 0.89, and 0.89, respectively. Similarly, for Folio dataset, *top-1*, *top-5* accuracies, *Pr*, *Re* and *F1* of CSTF with LDA classifier are 79.77%, 95.38%, 0.84, 0.80, and 0.79. For the Swedish leaf dataset, *top-1* and *top-5* accuracies for HM+HT with LDA classifier obtained are 92.01% and 99.7%, respectively. Other performance metrics like *Pr*, *Re*, and *F1* for Swedish leaf are 0.94, 0.93, and 0.93. For Leaf-12 dataset, the *top-1* accuracy, *top-5* accuracy, *Pr*, *Re*, *F1* obtained are 82.38%, 99.13%, 0.83, 0.83, and 0.83, respectively. The above metrics are achieved for a combination of CSTF features with RFC classifier.

Table 3.8 Summary of Results for PSR using Conventional Approach

Dataset	**Method**	**Accuracy**		**Precision, *Pr***	**Recall, *Re***	**F1-Score, *F1***
		Top-1	***Top-5***			
Flavia	CSTF+ LDA	89.17	98.96	0.91	0.89	0.89
Folio	CSTF+ LDA	79.77	95.38	0.84	0.80	0.79
Swedish leaf	HM + HT + LDA	92.01	99.7	0.94	0.93	0.93
Leaf-12	CSTF+ RFC	82.38	99.13	0.83	0.83	0.83

Most of the reported literatures utilize a preprocessing step (enhancement and segmentation) before the feature extraction and classification process. In this chapter, the reported results are obtained without preprocessing or enhancement techniques. The real-time dataset (Leaf-12) results are in correlation with the results of standard datasets (Flavia, Folio, Swedish leaf) inspite of the image complexity.

CSTF feature extraction method outperforms other combinations of features for the three datasets (Flavia, Folio, and Leaf-12). Also, it is observed

that the texture features contribute more to the classification of plant species. At the same time, both shape and color features are important for plant classification. In further chapters, the performance of the PSR system is improved by utilizing the Neural Network and Deep Learning methods.

CHAPTER 4

PERFORMANCE ANALYSIS OF PLANT SPECIES RECOGNITION SYSTEM USING NEURAL NETWORKS

4.1 INTRODUCTION

Artificial Neural Network (ANN) (Haykin 1999) is introduced to mimic the functionality of the human brain. There are different types of ANNs (Sivanandam & Deepa 2007), which are either based on supervised, unsupervised, or reinforced learning methodologies. In 1943, McCulloch and Pitts introduced the McCulloch and Pitt's (M-P) Neuron model. Hebb developed a Hebb network in 1949. In this network, the weight increases with the product of input and learning signal. Frank Rosenblatt introduced the concept of weight updation in Perceptron networks (feedforward networks). Network error minimization is demonstrated in Adaline networks in the year 1960. Rumelhart, Hinton, and Williams, in the year 1986, introduced the concept of multilayers and error factors in the hidden layer. The weight updation and minimization of error in each layer are explained using the Backpropagation algorithm.

In this chapter, the Backpropagation Neural Network (BPNN) with two hidden layers is investigated for Plant Species Recognition (PSR). Section 4.2 reported a survey on the usage of Neural Networks for PSR. Section 4.3 provides a detailed explanation of the proposed BPNN using the PReLU activation function. The results are discussed in Section 4.4. Section 4.5 concludes the chapter.

4.1.1 Structure of Biological Neuron

The human brain is a vital organ of the body that helps in receiving, processing, and transmitting information to other organs. There are about Quadrillion neurons in the human brain. It is highly complex in structure. A biological neuron transmits the information through electrochemical signaling on actuation. A single biological neuron is shown in figure 4.1. It contains the cell body called ‘Soma’. A nerve fiber that is connected to the soma is known as ‘Dendrites’. These dendrites possess many branches of nerve fibers. The information transfer from one neuron to another neuron is through the Axon. Many hair-like structures (strands) are present at the end of the axon. Every strand ends with a synapse. Synapses transmit the electric impulses to the adjacent neurons.

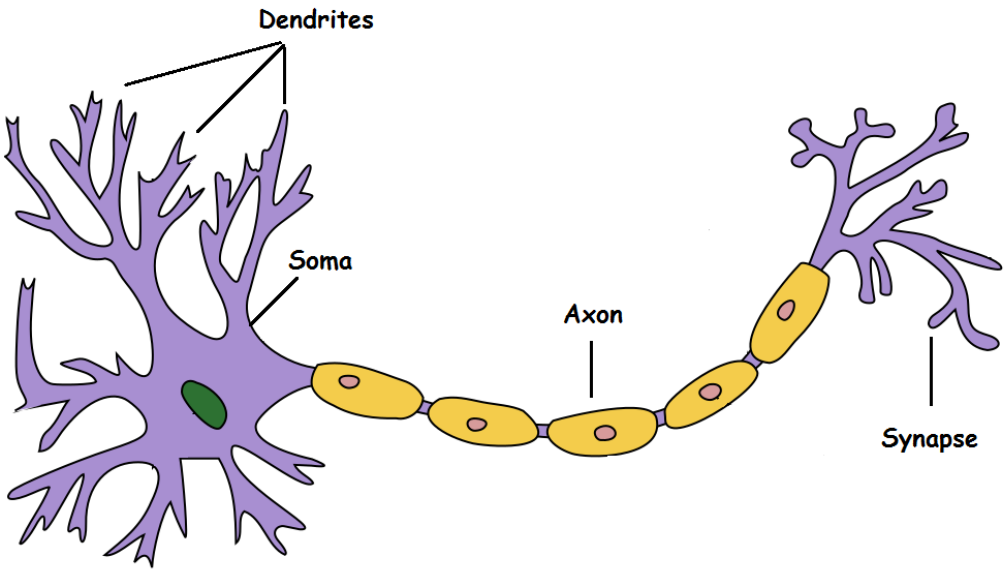


Figure 4.1 Biological Neuron (Source: www.towardsdatascience.com)

4.1.2 Artificial Neuron

An artificial neuron is the basic building block in Artificial Neural Network (ANN). An artificial neuron is based on a mathematical model. It is used to learn the data pattern or behavior. A neuron in the network is capable of accepting and processing the inputs. It is also capable of forwarding the

information to the adjacent neurons. These are termed as nodes in the network. In figure 4.2, the connections or arrows between the neurons is determined by its synaptic weights (w_1, w_2,..., w_m). The product of weights (w_1, w_2,..., w_m) and inputs (x_1, x_2,…,x_m), are summed along with a bias (b). Bias is a small value similar to that of weight. The activation function of a node is used to decide the progression or mitigation of the information from one neuron to the next neuron.

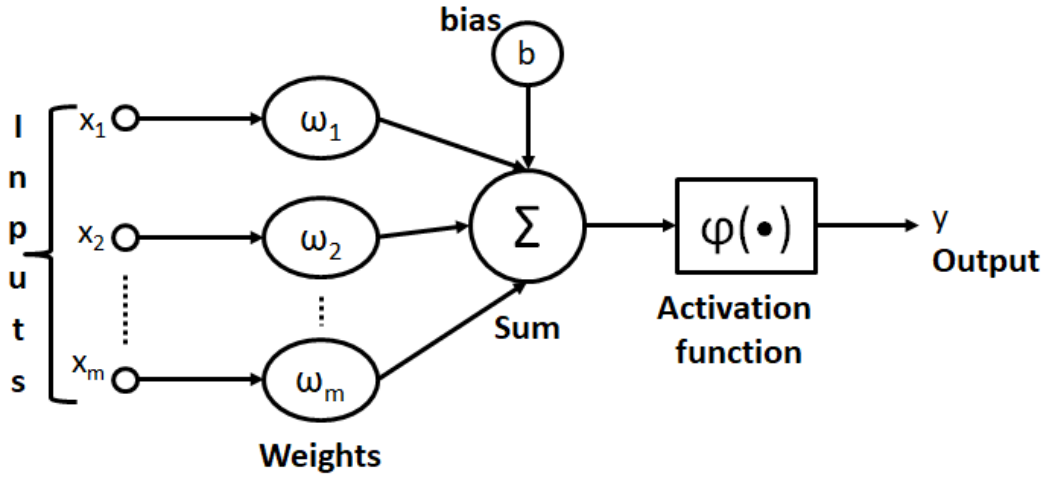


Figure 4.2 Structure of an artificial neuron (Source: www.medium.com)

4.1.3 Artificial Neural Network

An artificial neural network aims to identify the underlying association in a set of given data. It contains three important layers as specified below,

- Input layer
- Hidden layer
- Output layer

Figure 4.3 shows a simple ANN containing an input layer, two hidden layers, and an output layer. The input layer is provided with the input features or pixel values. The hidden layer processes the inputs and decides the

information transfer to the next layer. The number of output classes determines the neurons in the output layer. Input, hidden, and output layers are explained in section 4.1.3.1 to 4.1.3.3.

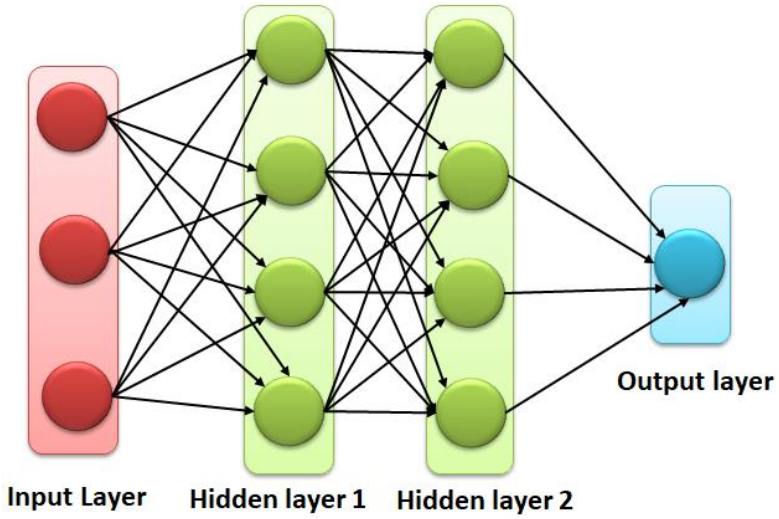


Figure 4.3 Artificial Neural Network with two hidden layers

4.1.3.1 Input layer

Inputs provided to the input layer neurons are in the form of extracted features or the raw pixel values. For example, in an image classification scenario, the pixel values (gray levels or color content) or the features (color or texture or shape or key-points or any combination of features) extracted from the image is given as an input data to the neural network model. The image data is flattened before it is passed on to the network. For example, the grayscale image of size 50 x 50 contains about 2500 pixel values. The pixel values are arranged in the form of a 1D vector with 2500 entries. Then the data (1D vector) is passed on to the neurons of the Input Layer. For a color image, the RGB channels are separated and flattened out before it is passed on to the neural network.

4.1.3.2 Hidden layers

The hidden layers form the central portion of the neural network model. It lies in between the input and output layers. The number of hidden layers and neuron/layer is decided by the programmer or designer of the network. The above-specified factors rely on the computational capability of the hardware in use and the application under development. The IN and OUT connections of the neurons in the hidden layer are associated with the synaptic weights. The weights specify the importance of the connections. Initially, the weights are assigned in a random fashion. Later, the weights are periodically updated to minimize the loss parameter of the network. The weight updation ceases after reaching certain stopping criteria such as attaining a minimum loss or number of epochs.

4.1.3.3 Output layer

The final layer of the neural network is termed as an Output Layer. The number of neurons in the output layer is determined by the number of output classes. The output layer neurons are provided with an activation function such as Softmax or Sigmoid functions. The most frequently utilized activation functions are Sigmoid and ReLU (Rectified Linear Unit). The details about the activation functions are described in subsection 4.3.3. The network loss parameter (Error in prediction) is computed in the output layer, using different methods such as Categorical Cross-Entropy Loss, Mean Square Loss, and Hinge Loss functions.

4.2 SURVEY ON NEURAL NETWORKS FOR PLANT SPECIES RECOGNITION

This section details the recent literatures on plant species recognition utilizing Neural Networks.

Pacifico *et al.* (2018) examined the performance of five different classifiers, namely, K-Nearest Neighbor (K-NN), Decision Tree (CART), Naïve Bayes (NB), Support Vector Machine (using two-kernel functions-linear and Radial Basis Function (RBF)) and Multi-layer Perceptron (MLP) with Backpropagation algorithm. Iris, wheat seeds, and 100 plant leaves datasets are utilized for testing the performance of the classifiers. MLP with Backpropagation algorithm outperformed the other four classifiers.

Adinugroho & Sari (2018) proposed a fusion of color, shape, and texture features (31-d) for the classification of leaves. Swedish leaf images are segmented and utilized for classification. Out of 31 features, fifteen features are obtained by morphological operations on segmented leaves. Remaining features are obtained by using Hu moments, Gray Level Co-occurrence Matrices (GLCM), and Color Channel Statistics (12-d using mean, standard deviation, skewness, and kurtosis). The extracted features are sorted based on the Pearson Correlation Coefficient and passed onto the Neural Network for training. A Feedforward Neural Network (FFNN) (input layer- 32 neurons, single hidden layer-3 neurons, and output layer-13 neurons) with Backpropagation is utilized for classification. The Hyperbolic tangent sigmoid activation function is used in the hidden and output layers of the Neural Network. The authors reported that the utilization of 23 features (10 shape features, 9 color features, and 4 texture features) and 14 neurons in the hidden layer resulted in improved performance for plant species recognition system.

Kho *et al.* (2017) proposed a framework for identifying the three Ficus species of identical morphology. 54 herbarium specimens are utilized to capture the 60 Ficus leaf images. After applying a series of preprocessing methods, the feature extraction is performed. Features are extracted using five shape features, Hu moments, GLCM, and Histogram of Oriented Gradients (HoG). Then, the extracted features are classified using Artificial Neural

Network (ANN- 1 Hidden layer with 10 neurons) and Support Vector Machine (SVM). Artificial Neural Network performed better in the classification of Ficus plant species (Accuracy of 83.3%) compared to Support Vector Machine.

Wang *et al.* (2016) utilized two techniques for plant species recognition, namely, Pulse-Coupled Neural Network (PCNN) (Wang *et al.* 2010) and Support Vector Machine (SVM). An important feature termed as an Entropy Sequence is acquired from PCNN. Additional features are obtained from morphological and mathematical methods. These features are classified with the help of SVM, a multi-class classifier utilizing the Gaussian Radial Basis Function. Flavia, ICL, and MEW2012 datasets are used to test the methodologies.

Wable & Chilveri (2016) proposed an Artificial Neural Network (ANN) termed as Pulse-Coupled Neural Network (PCNN) for plant classification. The features from the leaf images are extracted by using Zernike Moment, Hu Invariants, Entropy Sequence, and morphological features. The extracted features are used as inputs to the Artificial Neural Network (1 - Hidden Layers with 20 neurons). The proposed architecture is evaluated by using the Flavia dataset.

Codizar & Solano (2016) proposed a leaf recognition system termed as LeaVes utilizing the leaf shape features and vein patterns. Shape features are obtained by using Canny edge detection, Centroid-Radii model, and Hu moment. Leaf vein features are extracted by utilizing the morphological operation (opening) and image differencing technique. The extracted features act as input nodes of Multilayer Perceptron (MLP). MLP consists of an input layer (189 nodes), a hidden layer (147 nodes), and an output layer with 32 nodes (representing the number of classes).

Sharma & Gupta (2015) proposed a multi-layer feedforward Neural Network for the classification of 16 different plant species (440 leaf images). Leaf images are pre-processed before extracting the 12 morphological features. Then, the features are supplied as inputs to the Neural Network. To the twelve features, the Neural Network with Backpropagation algorithm is applied. The Neural Network comprises of four layers (input layer-12 neurons, 2 hidden layers with 90 neurons/layer, output neuron-1).

Aakif & Khan (2015) proposed a method for classifying the plant species using Artificial Neural Network (ANN). In the pre-processing stage, the leaf portion in the image is segmented and normalized. The features are extracted by using morphological features, Fourier Descriptors, and Shape-defining methods. These extracted features act as an input to the ANN. The ANN with Backpropagation algorithm is trained for 1000 iterations. In the output layer of ANN, the value of mean squared error is noted for each iteration. The authors tested the ANN architecture on Flavia, ICL dataset, and Self-collected datasets.

Chaki *et al.* (2015) utilized the fusion of texture and shape features for the classification of plant species. Gabor filter coefficients (Weldon *et al.* 1996) and the Gray Level Co-occurrence Matrices (GLCM) are extracted (leaf texture features). Shape features are extracted by using Curvelet Transform and Invariant Moments. Neuro-Fuzzy Controller (NFC) and Multilayer Perceptron (MLP) with Backpropagation algorithm are used in the classification of plant species.

Deokar *et al.* (2013) proposed a framework for leaf recognition and extracted two sets of features. One set comprised of 28 feature points and another with 60 feature points. For classification, a multi-layer feedforward ANN with Backpropagation algorithm is used. It is trained with Gradient

Descent optimizer. The authors reported that the leaf recognition rate is greatly affected by increasing the number of features for classification.

Janani & Gopal (2013) proposed a fusion of color, shape, and texture features for plant classification by Artificial Neural Network (ANN). Shape features are obtained from the leaf geometry (Compactness, Eccentricity, Aspect Ratio), and Hu Moments. Color features are extracted by utilizing the Color moments (Mean, Standard Deviation, Skewness, and Kurtosis). Gray Level Co-occurrence Matrix (GLCM) is used to extract the texture features. A single hidden layer with 10 neurons is used in ANN. The author reported that a minimum of eight features is required to achieve better performance in classification.

Rankothge *et al.* (2013) developed an Advanced Plant Identification System (APIS) based on leaf photographs. The dataset is created from three different versions of the same leaf photographs. The first version includes the binary thresholded images of a leaf. In the second version, the leaf texture features are extracted using the Sobel algorithm, and the thickening of edges is carried out using dilation operation. The third version includes cumulative RGB color values. The images obtained from the three versions are converted to matrix values. The matrix values are made rotation invariant by using the 2D-FFT (Fast Fourier Transform). The 2D-FFT coefficients are fed as an input to the Neural Network with Backpropagation algorithm.

Clark *et al.* (2012) proposed a Multilayer Perceptron based methodology to identify the plant species of the same genus comprising of 516 images. The plant species (*Tilia Cordata, Tilia Americana, Tilia Platyphyllos,* and *Tilia Tomentosa*) considered in the studies are obtained from Royal Botanical Garden, United Kingdom. Twenty-two morphological characteristics are extracted from the leaf images and fed as an input to the Neural Network. The Neural Network is constructed with one hidden layer, where the number of

neurons in the hidden layer is varied between 24 to 96. The author reported that the best result in classification is achieved when the number of neurons in the hidden layer is about 54. Also, the learning rate of the Neural Network is optimized for better classification of plant species. Classification accuracy of about 44% is achieved after optimizing the learning rate and the number of neurons in the hidden layer. The author also suggested that the accuracy could be improved further by increasing the number of images in the dataset and by utilizing the dimensionality reduction techniques.

Kadir *et al.* (2012) demonstrated the leaf recognition system based on Principal Component Analysis (PCA) and Probabilistic Neural Network (PNN). Shape (morphological features), texture (GLCM), color (RGB Color moments), and vein patterns (obtained from the morphological operation) are extracted from the leaf images and fed into PCA. PCA resulted in a set of orthogonal features. These features are then classified using the PNN. Flavia and Foliage datasets are used for analyzing the performance. The authors reported that the utilization of 25 features with PCA resulted in better classification accuracy compared to 54 features without PCA method.

Chaki & Parekh (2011) proposed a feature extraction method using Moment Invariants (M-I) and Centroid-Radii (C-R) shape models. It resulted in a 38-d feature vector. These features are provided as an input to the Neural Network with a structure of 38 (Input layer neurons) - 30 (Hidden layer neurons) - 3 (Output layer neurons). The dataset contains three classes with 60 images per class. The convergence on training the neural network occurred at 20706 epochs.

Zulikifli *et al.* (2011) proposed a method for plant species recognition. Moment Invariant Techniques such as Zernike Moment Invariant, Legendre Moment Invariant, and Tchebichef Moment Invariant are used for feature extraction. Classification is carried out using General Regression Neural

Network (GRNN)). Best classification accuracy is obtained by using a combination of Tchebichef Moment Invariant feature extraction method and GRNN.

Fu *et al.* (2006) utilized the leaf vein patterns for the classification of plant species. Segmentation and thresholding techniques are performed on the leaf images to extract the leaf vein patterns. Then, ten different features are extracted from the vein patterns and passed as inputs to the Artificial Neural Network (1 hidden layer with 20 neurons) for classification.

Gu *et al.* (2005) proposed a leaf segmentation method by employing the Wavelet Transform and Gaussian Interpolation methods. From the segmented leaf portion, the Run Length (RL) features are extracted and classified using different methods. The classifiers considered are Nearest Neighbor classifier (1-NN), K-Nearest Neighbor classifier (K-NN), and Radial Basis Probabilistic Neural Network (RBPNN). The dataset contains about twenty different plant species. The highest classification accuracy of 93.17% is obtained with the Nearest Neighbor classifier (1-NN).

Table 4.1 summarizes the literature survey on Plant Species Recognition using Neural Networks. It is observed that most of the reported literatures on plant species recognition employed Neural Networks with a single hidden layer. Also, the features are extracted before it is passed as inputs to the Neural Networks for classification. The reported Neural Networks supported the Backpropagation algorithm. In few literatures, the Probabilistic Neural Network (PNN) and General Regression Neural Network (GRNN) are also reported for the PSR system. Sigmoid and the ReLU activation functions are commonly utilized in the reported literatures. The classification accuracies obtained by the above-specified methods could be further improved by adding more number of hidden layers, increasing the number of neurons/layer and PReLU activation function.

Table 4.1 Summary of reported literature on Plant Species Recognition using Neural Networks

S. No.	Feature Extraction			Classifier	Author & Year
	Shape Features	Texture Features	Color Features		
1.	Shape features and other features varying for each dataset		-	KNN, CART, NB, SVM, and MLP	Pacifico *et al.* (2018)
2.	15 morphological features and Hu moments	GLCM	Color Moments	FFNN	Adinugroho & Sari (2018)
3.	5 shape features, Hu moments	GLCM, HoG	-	ANN and SVM	Kho *et al.* (2017)
4.	Zernike moment, Hu invariants	Entropy sequence from PCNN	-	SVM	Wang *et al.* (2016)
5.	Zernike moment, Hu invariants, and other morphological features	entropy sequence (PCNN)	-	ANN	Wable & Chilveri (2016)
6.	Canny edge detection, Centroid-Radii model and Hu moment	Vein features	-	MLP	Codizar & Solano (2016)
7.	12 morphological features	-	-	FFNN	Sharma & Gupta (2015)
8.	morphological features, Fourier descriptors, and new Shape-defining features	-	-	ANN	Aakif & Khan (2015)
9.	Curvelet transform, Invariant Moments	Gabor filter, GLCM	-	NFC and MLP	Chaki *et al.* (2015)
10.	Geometric features	-	-	Multi-layer feedforward ANN	Deokar *et al.* (2013)
11.	Geometric features (compactness, eccentricity, aspect ratio) and Hu moments	GLCM	Color Moments	FFNN	Janani & Gopal (2013)

Table 4.1 (Continued)

S. No.	Feature Extraction			Classifier	Author & Year
	Shape Features	Texture Features	Color Features		
12.	2D-FFT			ANN	Rankothge *et al.* (2013)
13.	22 Morphological characters	-	-	MLP	Clark *et al.* (2012)
14.	Morphological features	Vein features, GLCM	Color moments	PNN	Kadir *et al.* (2012)
15.	M-I and C-R shape models	-	-	Neural Network	Chaki & Parekh (2011)
16.	ZMI, LMI and TMI	-	-	GRNN	Zulikifli *et al.* (2011)
17.	-	10 vein features	-	ANN	Fu *et al.* (2006)
18.	-	RL features	-	1-NN, k-NN, RBPNN	Gu *et al.* (2005)

4.3 PROPOSED METHODOLOGY FOR PLANT SPECIES RECOGNITION USING NEURAL NETWORK

A four-layered Artificial Neural Network is constructed and tested for Plant Species Recognition. The four layers are in the form of one input layer, two hidden layers, and one output layer. Four datasets (Flavia, Folio, Swedish leaf, and Custom-developed Leaf-12) are utilized to evaluate the proposed Neural Network for its classification accuracy. The details about the pre-processing of images, Neural Network architecture, and performance analysis of the network, are described in the following subsections.

4.3.1 Image Preprocessing and Normalization

Images in the dataset are resized to 32x32x3 using a Nearest Interpolation method. Image resizing is performed by maintaining the aspect ratio. Then, the 3072 (32x32x3) pixel intensities are normalized (Range of

values lies between 0 and 1). The normalized pixel intensities are given as an input to the proposed Neural Network.

4.3.2 Backpropagation Neural Network Algorithm

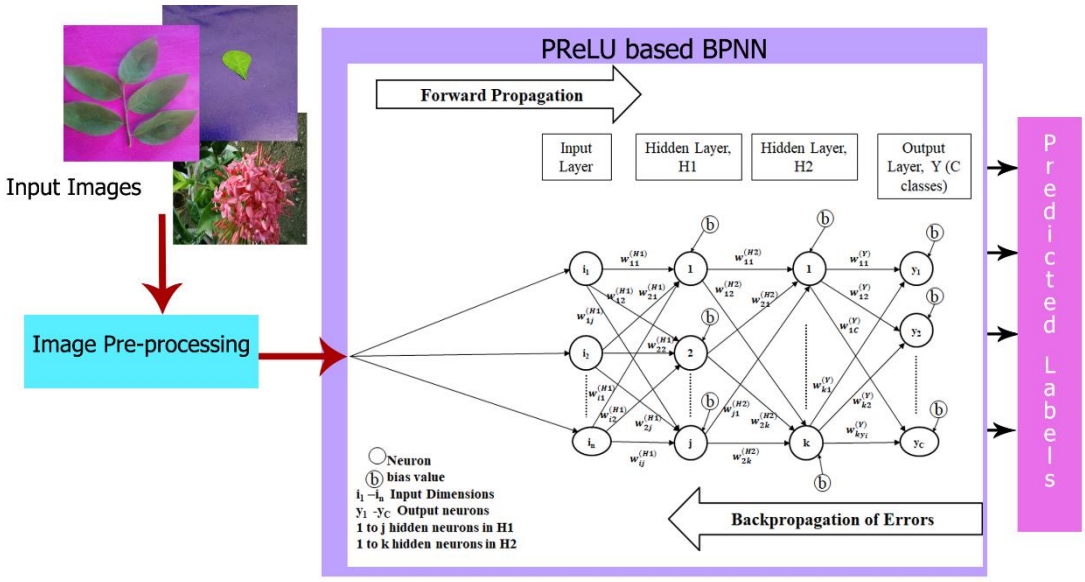


Figure 4.4 Architecture on PReLU based Backpropagation Neural Network for Plant Species Recognition

The proposed architecture on PReLU based Neural Network with Backpropagation algorithm is shown in figure 4.4. The leaf images from the dataset (Flavia, Folio, Swedish leaf, and Leaf-12) are pre-processed (resized to 32x32x3 by maintaining the aspect ratio, and the pixel intensities are normalized) before it is fed as an input to the proposed Neural Network. The Neural Network consists of four layers, namely, one input layer, two hidden layers, and an output layer. In the input layer, there are about 3072 neurons (32 x 32 x 3 = 3072). The number of neurons in the first and second hidden layer is about 500 (H1) and 250 (H2), respectively. The hidden layer neurons utilize the PReLU based activation function to decide on the transfer of information from one neuron to the adjacent layer neuron.

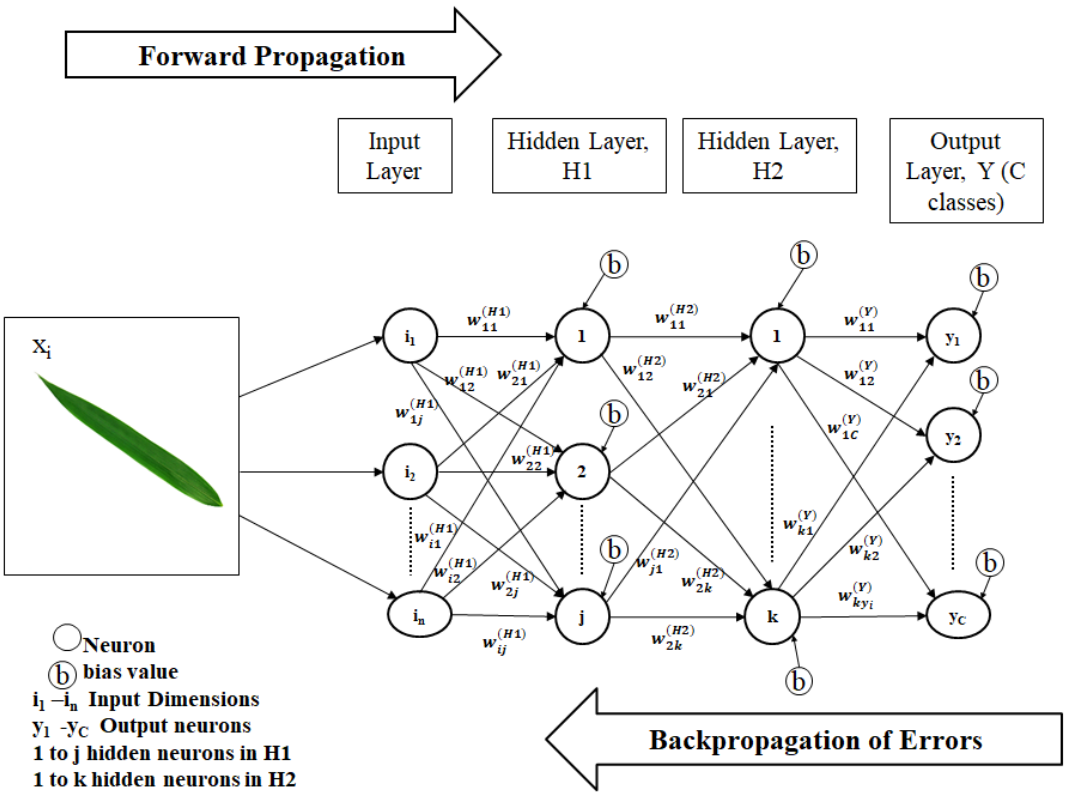


Figure 4.5 Methodology on PReLU based Neural Network with Backpropagation Algorithm

The number of neurons in the output layer is determined by the number of classes in the dataset. For Flavia and Folio datasets, the number of output layer neurons is about 32. For Swedish leaf and custom-developed Leaf-12 dataset, the number of neurons in the output layer is 15 and 12, respectively. The predictions are obtained in the output layer, resulting in the estimation of the loss function. The errors are backpropagated through the network, resulting in the updation of weights in each of the connections to minimize the loss function. When the loss function attains a low value, the weight updation in each of the connection gets terminated. The methodology and algorithm depicting the proposed Neural Network with Backpropagation algorithm are depicted in figure 4.5 and figure 4.6 (Sivanandam & Deepa 2007), respectively.

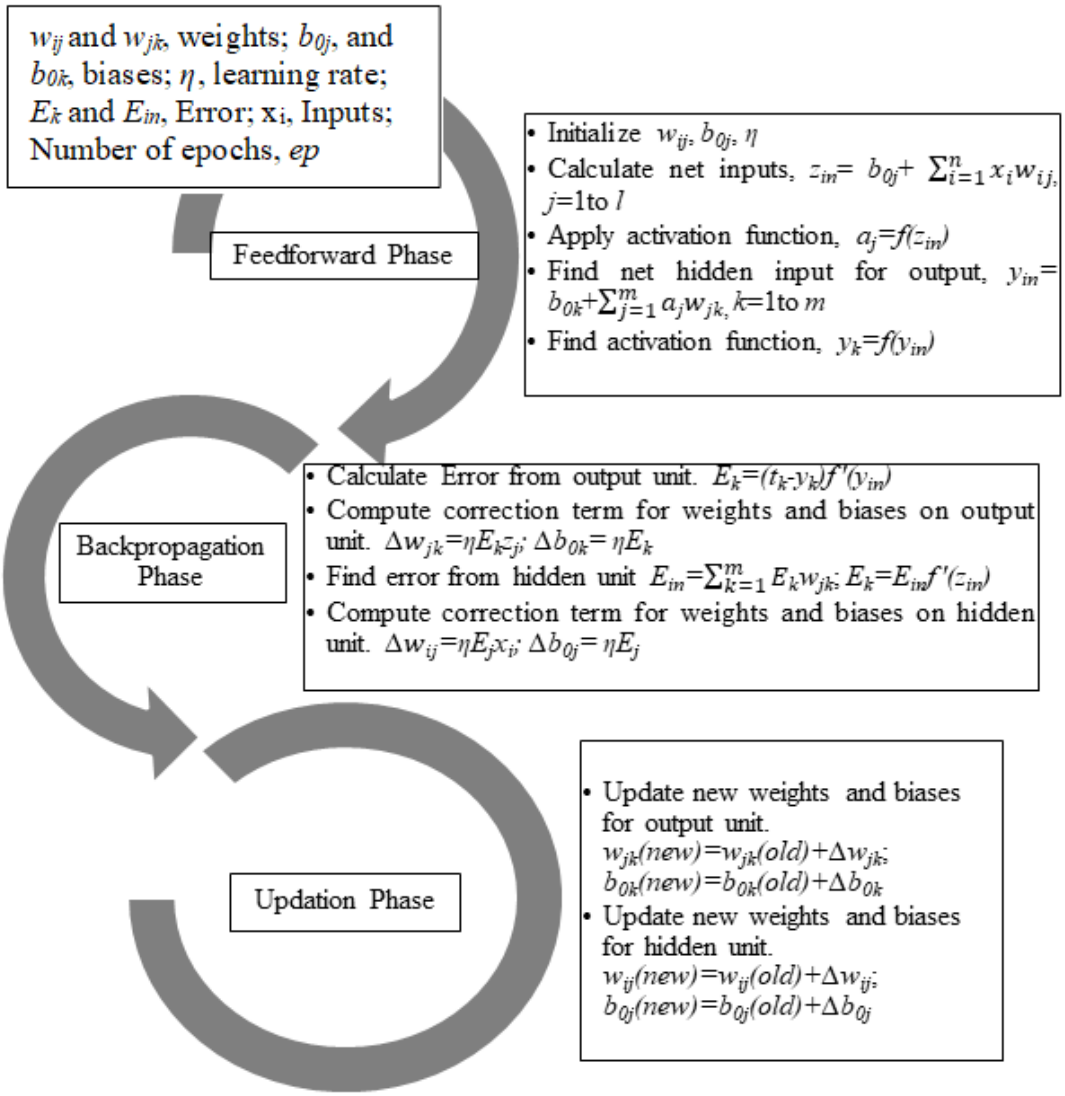


Figure 4.6 Steps involved in PSR system by using PReLU based Neural Network with Backpropagation algorithm

Neural Network with Backpropagation algorithm has three execution phases, namely, the Feedforward phase, Backpropagation phase, and Updation phase. During the feedforward phase, the net inputs (z_{in}) are calculated. Then, the activation function is tested on the estimated net input to determine the possibility of information transfer to the next neuron. In the backpropagation phase, the errors are calculated from the output layer to the input layer. New weights and biases are updated in the updation phase.

4.3.3 Activation Function

Activation function plays a major role in Neural Networks. It determines to propagate or mitigate the information transfer from one neuron to the adjacent layer neuron. In other words, an activation function is responsible for actuating the summed inputs from the neurons of one layer to the next one. Several activation functions are reported in the literatures. Some of them are Sigmoid activation function (Sivanandam & Deepa 2007), Tanh activation function (Sivanandam & Deepa 2007), Rectified Linear Unit (ReLU) (Goodfellow *et al.* 2016), Parametric ReLU (PReLU) (He *et al.* 2015), Exponential Linear Unit (ELU) (Clevert *et al.* 2015) and Leaky ReLU (LReLU) (Maas *et al.* 2013).

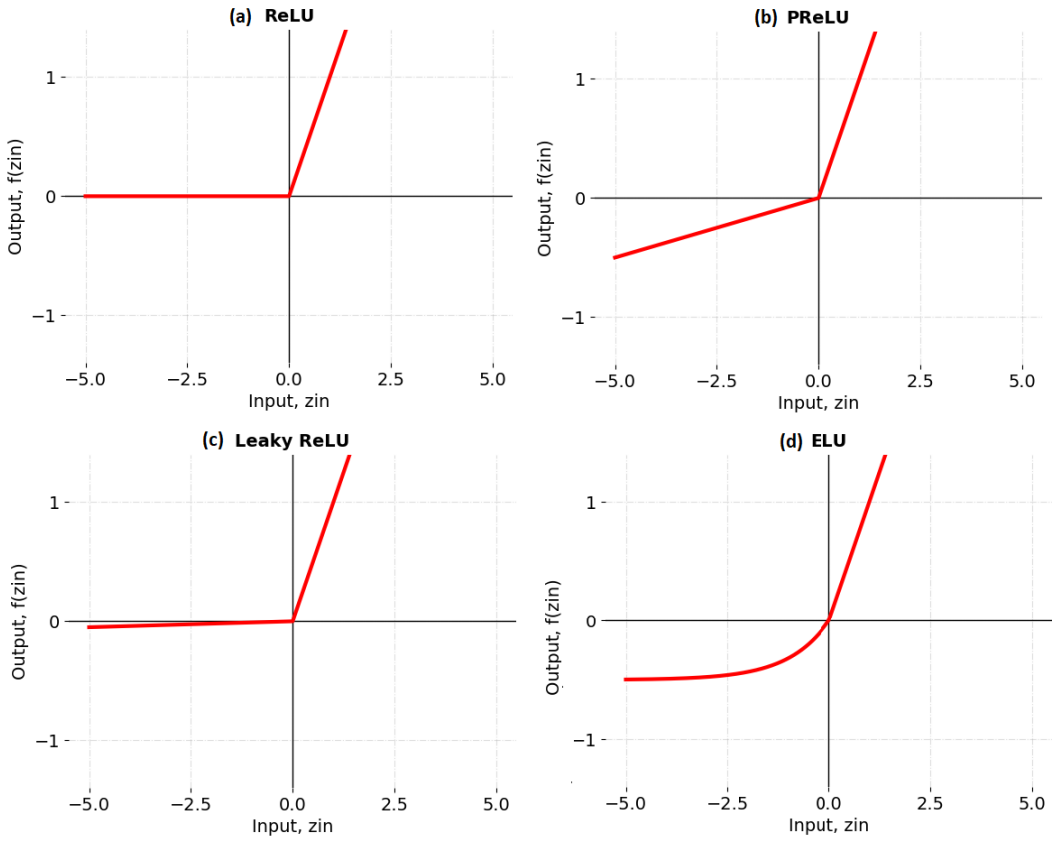


Figure 4.7 Representation of Activation Functions

ReLU based activation functions are detailed in the following subsections. ReLU activation function is most widely used in the Neural

Networks because of its unity slope in the first quadrant (Positive values of *x* and *y*). But, it introduces dead neurons in the Neural Network if the net input (z_{in}) falls below 0. Other modified versions of ReLU activation functions (PReLU, ELU, and LReLU) are introduced to utilize the neurons in the network efficiently. Figure 4.7 shows the pictorial representation of the four ReLU based activation functions.

4.3.3.1 ReLU

Rectified Linear Unit (ReLU) is a non-linear activation function (Goodfellow *et al.* 2016). For the positive values of z_{in}, the function outputs the maximum value between z_{in} and 0. For the negative values of z_{in}, the function outputs the value of zero (reduces the information transfer). It is represented in Equation (4.1). Since the negative values of z_{in} do not contribute to the final outcome, it results in a large number of dead neurons in the network. Dead neurons in the Neural Network are minimized by the modified versions of ReLU activation functions (PReLU, ELU, and LReLU).

$$f(z_{in}) = \begin{cases} \max(z_{in}, 0), & if\ z_{in} > 0 \\ 0, & if\ z_{in} \leq 0 \end{cases} \tag{4.1}$$

4.3.3.2 PReLU

He *et al.* (2015) introduced a Parametric Rectified Linear Unit (PReLU), which is an enhancement of the ReLU activation function. The number of dead neurons in the Neural Network is minimized by utilizing the PReLU activation function. For positive values of z_{in}, the determination of output is similar to ReLU activation functions. For the negative values of z_{in}, the output is determined by the product of z_{in} and α_i, as represented in Equation (4.2). The negative portion's slope is regulated by α_i.

$$f(z_{in}) = \begin{cases} z_{in}, & if\ z_{in} > 0 \\ \alpha_i z_{in}, & if\ z_{in} \leq 0 \end{cases} \tag{4.2}$$

4.3.3.3 Leaky ReLU

Leaky ReLU (LReLU) is similar to PReLU activation function. Maas *et al.* (2013) introduced this activation function to minimize the dying neurons when z_{in}<0. For LReLU, the α_i value in PReLU activation function is set to 0.01 when z_{in} <0, as given in Equation (4.3).

$$f(z_{in}) = \begin{cases} z_{in}, \ if \ z_{in} > 0 \\ 0.01 * z_{in}, \ if \ z_{in} \leq 0 \end{cases} \tag{4.3}$$

4.3.3.4 ELU

Exponential Linear Unit (ELU) (Clevert *et al.* 2015) solves the Vanishing Gradient problem. When the neural network training encompasses gradient based learning and backpropagation, Vanishing Gradient issue arises. The gradient value decreases exponentially (vanishingly small value) as backpropagation process is performed from the final layer to the initial layer. Sometimes vanishingly small gradient value prevent weight updation process from changing weight value. This is termed as Vanishing Gradient problem. ELU has a faster convergence. It is slow to train as it requires the exponential values of net inputs (z_{in}) to determine the output when z_{in} <0. The ELU activation function is represented in Equation (4.4).

$$f(z_{in}) = \begin{cases} z_{in}, \ if \ z_{in} > 0 \\ \alpha(\exp(z_{in}) - 1), if \ z_{in} \leq 0 \end{cases} \tag{4.4}$$

4.3.4 Loss Function

The output layer of the Neural Network utilizes the loss function to determine the performance of the system. The loss function provides a relation between the predicted output and the ground truth value. It is termed as an objective function or error function or cost function of the network. Minimization of the loss function (Categorical Cross-Entropy loss function, E_{CC}) is carried out in the Backpropagation phase of the PReLU based Neural

Network. Categorical Cross-Entropy (E_{CC}) (Rusiecki 2019) loss function is represented in Equation (4.5). This loss function is widely used in Neural Networks with multi-class prediction. In figure 4.6, E_{CC} is utilized in place of the loss functions (E_k and E_{in}) used in the Backpropagation phase of PReLU based Neural Networks.

$$E_{CC} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} l_{ic}(\log(Y_{ic})) \qquad (4.5)$$

Here, N is the number of training sets. These training sets vary between (x_1, l_1) to (x_N, l_N), where x_i represents an input vector, and l_i denotes a target vector of c (c=1 to C) categories. y_i represents the predicted label. l_{ic} represents binary indicator function for the i^{th} training sample, belonging to c category. $Y_{ic}\epsilon c$ is the predicted probability for the i^{th} sample.

4.3.5 Training of Algorithm

Training a network is a process in which the Neural Network is exposed to the dataset images, resulting in the prediction of a label. The training process is terminated under certain stopping criterion as given below,

1. low loss in prediction

2. number of epochs

During the updation phase of P-BPNN, the connection weights are updated. The rules to specify the weight updation process are termed as optimizers. The most frequently used optimizers are Stochastic Gradient Descent (SGD), Adam, Adamax, and Adadelta. These are discussed in the following subsections 4.3.5.1-4.3.5.4.

4.3.5.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is the traditional weight updation rule, represented in Equation (4.6) (Ruder 2016). The weight updation is related

to the learning rate (η), cost function ($J(w)$) for each data, $x_{(i)}$, and label $y_{(i)}$. w_{old} is the old weight, and w_{new} is the updated weight.

$$w_{new} = w_{old} - \eta . \nabla_w J(w; x_{(i)}; y_{(i)}) \quad (4.6)$$

4.3.5.2 Adam

Adam (Ruder 2016; Kingma & Ba 2014; Brownlee 2017) is an effective stochastic optimization method. It is based on the estimation of the adaptive moment. It inherits the properties of Adagrad and RMSprop (Root Mean Square gradient) optimizers. It is given by,

$$w_{new} = w_{old} - \eta . \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon} \quad (4.7)$$

Here, $\widehat{m_t}$ and $\widehat{v_t}$ are bias-corrected first and second moments, both dependent on β_1=0.9, β_2=0.999. w_{new} and w_{old} are newly updated and old weights. Learning rate or stepsize is represented as, η and ϵ=10^{-8}.

4.3.5.3 Adamax

Adamax (Ruder 2016; Kingma & Ba 2014) is an extended version of Adam optimizer, represented in Equation (4.8). It is related to the η (learning rate), u_t (infinity norm exponentially weighted) and $\widehat{m_t}$ (bias-corrected first moment).

$$w_{new} = w_{old} - \frac{\eta}{u_t} . \widehat{m_t} \quad (4.8)$$

4.3.5.4 Adadelta

Adadelta (Zeiler 2012; Ruder 2016) is introduced to address the issue of decreasing learning rate in the Adagrad optimizer. The weight updates are based on the sliding window. In Adadelta optimizer, fixed window size 'w_i' is used. Adadelta optimizer rule is given in Equation (4.9).

$$w_{new} = w_{old} + \left(-\frac{\eta}{RMS[g]_t}\right).g_t \qquad (4.9)$$

Here, g_t is the gradient. The Root Mean Square (RMS) error criteria is given by, $RMS[g]_t = \sqrt{E[g^2]_t + \epsilon}$, where ϵ is a constant. $E[g^2]_t$ is an exponentially decaying average of the squared updated parameter.

4.4 Results and Discussion

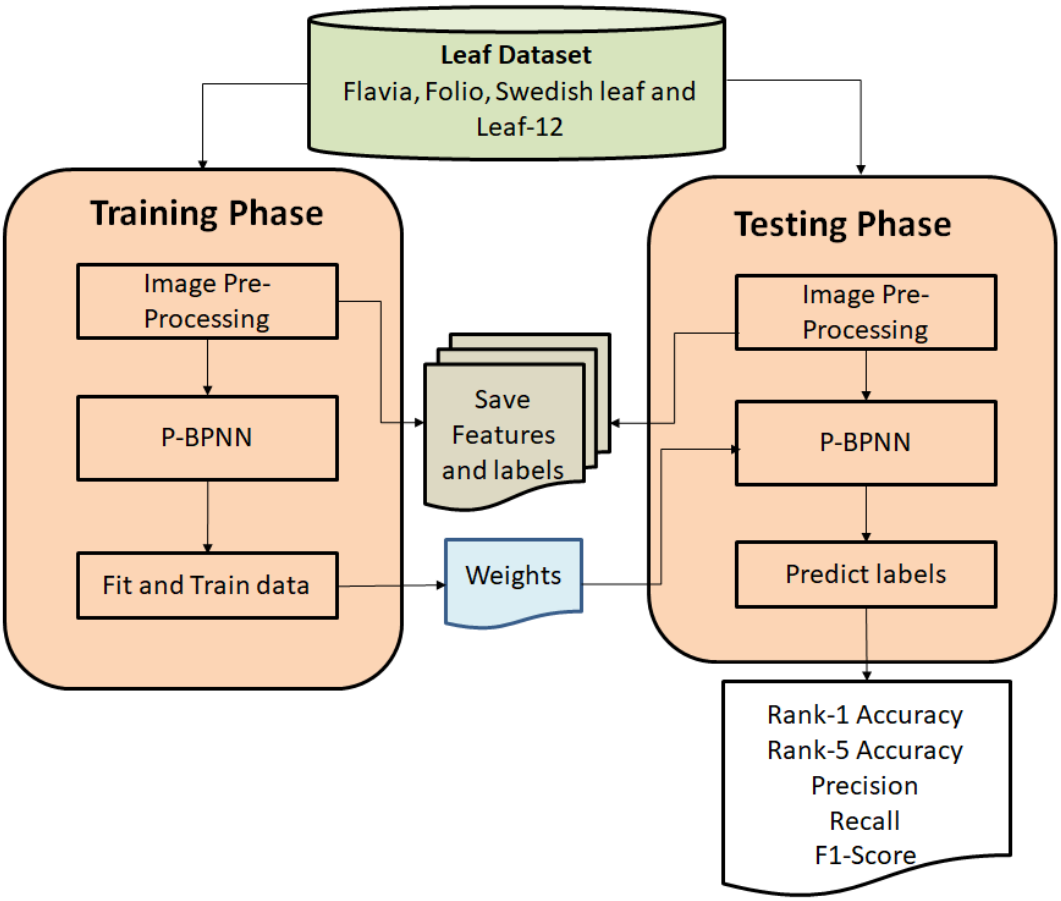


Figure 4.8 Steps involved in the Experimentation Procedure of PReLU based Backpropagation Neural Network (P-BPNN)

Performance analysis of proposed Neural Network with Backpropagation algorithm for Plant Species Recognition is carried out. Four datasets (Flavia, Folio, Swedish leaf, and Custom-developed Leaf-12) are considered. Implementation is performed on an i7 processor with 20GB RAM.

The processor is also connected to the NVIDIA Titan X GPU through the PCI interface. Software packages utilized for the implementation are Python 3.5 with supporting libraries like Keras with Tensorflow as Backend, Scikit-learn, Hdf5, OS, Matplotlib, and Seaborn. The procedure for experimentation is detailed in the form of a flowchart in Figure 4.8.

The Performance analysis is carried out by setting few parameters to constant values (Number of layers in the architecture = 4 including 2 Hidden layers, Number of neurons in Hidden layer-1 = 500, Number of neurons in Hidden layer-2 = 250, Batch size = 32, Train/Test data split ratio = 70:30). The accuracy of the PSR system is obtained by varying different hyperparameters such as Optimizers, Learning rate, Activation function, Epochs, and Alpha value in the PReLU activation function.

4.4.1 Flavia dataset

Table 4.2 Identification of best Hyperparameters for P-BPNN

Optimizers	**Accuracy (%)**	
	Top-1	**Top-5**
SGD	77.71	97.29
Adam	**82.50**	**98.75**
Adamax	80.62	97.71
Adadelta	77.92	97.29
Learning rate		
0.001	**82.50**	**98.75**
0.0001	77.08	97.50
0.00001	65.42	94.79
Activation Function		
ReLU	80.21	97.29
PReLU	**82.50**	**98.75**
ELU	80.21	97.29
Leaky ReLU	77.50	97.50

The appropriate selection of hyperparameters (optimizers, learning rate, and activation function) is carried out on the Flavia dataset. Table 4.2 displays the *top-1* and *top-5* accuracies obtained by varying the

hyperparameters. It is observed that Adam optimizer performed better compared to other optimizers (Stochastic Gradient Descent, Adamax, and Adadelta). The reported accuracies are obtained for 50 epochs.

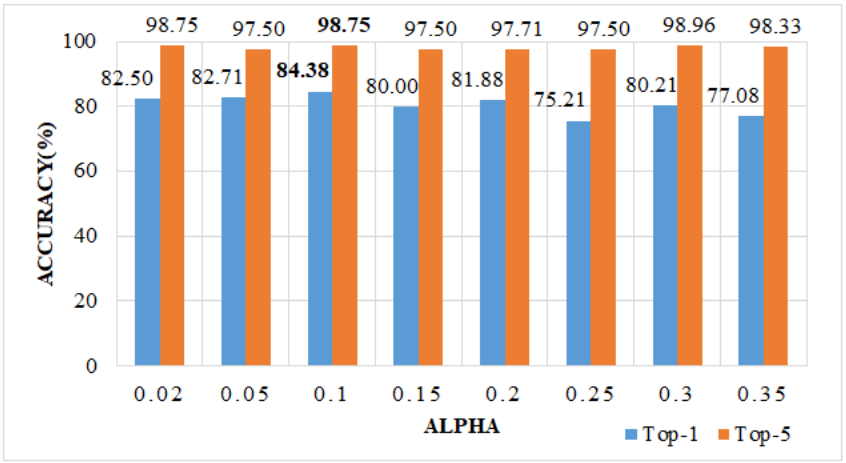
The accuracy of Adam optimizer is greater than SGD, Adamax, and Adadelta optimizer by 4.79%, 1.88%, and 4.58%, respectively. Adam optimizer outperforms other optimizers, and this might be due to its capability of adaptively learning data faster with the help of gradient moments.

Further, the learning rate (0.001 to 0.00001) is varied to determine the accuracy of the PSR system (Adam optimizer, 50 epochs). From Table 4.2, it is observed that the learning rate of 0.001 resulted in higher accuracies (*Top-1*: 82.50%, *Top-5*: 98.75%) compared to other learning rates (0.0001, 0.00001). Increasing the learning rate from 0.001 to 0.01 resulted in a higher loss. Hence, the performance result for the learning rate of 0.01 is not included in Table 4.2.

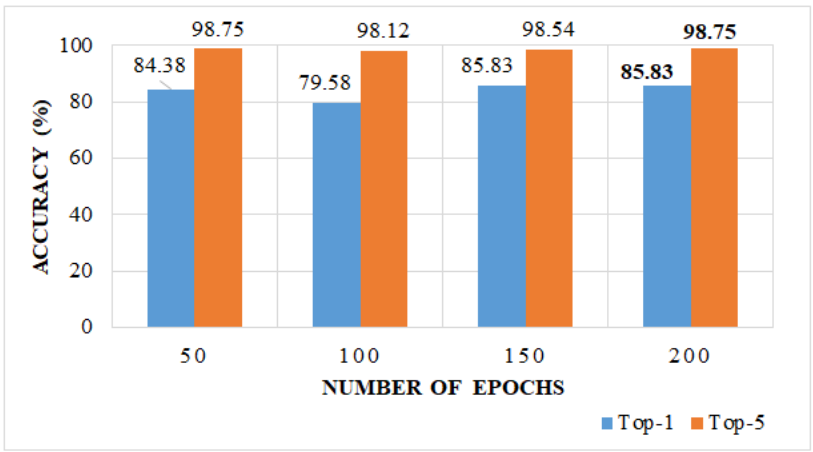
Later, the performance analysis is carried out by varying the activation functions. The hyperparameter settings of proposed Neural Network are Adam optimizer, a learning rate of 0.001, and 50 epochs. It is observed that the PReLU activation function for the proposed architecture resulted in higher *Top-1* (82.5%) and *Top-5* (98.75%) accuracies. This is due to the inclusion of negative inputs using the alpha hyperparameter (α = 0.02) in the PReLU activation function.

Further, the optimization of α is carried out by maintaining hyperparameter settings (Adam optimizer, Learning rate = 0.001, Number of epochs = 50, PReLU activation function). The α value is varied from 0.02 to 0.4. The accuracies obtained by varying the α are shown in figure 4.9(a). It is observed that the top-1 accuracy of 84.38% resulted when the α value is fixed at 0.1. Later, the number of epochs is varied to determine the best accuracy for the PSR system, as shown in figure 4.9(b). These hyperparameters (Adam optimizer, Learning rate = 0.001, PReLU activation function, α = 0.1) is

maintained during the analysis. It is observed that the highest accuracy (*Top-1*: 85.83%, *Top-5*: 98.75%) is obtained at the 200th epoch. The confusion matrix obtained for the appropriate parameters of the proposed Neural Network architecture is shown in figure 4.10.



(a)



(b)

Figure 4.9 Analysis of Flavia dataset for P-BPNN (a) Determination of Alpha (α) value for PReLU activation function (b) Number of epochs

To summarize, the proposed Neural Network architecture with Backpropagation algorithm performs better compared to other reported literatures for the PSR system, as shown in figure 4.11. The proposed architecture performs better with the hyperparameter settings such as Adam

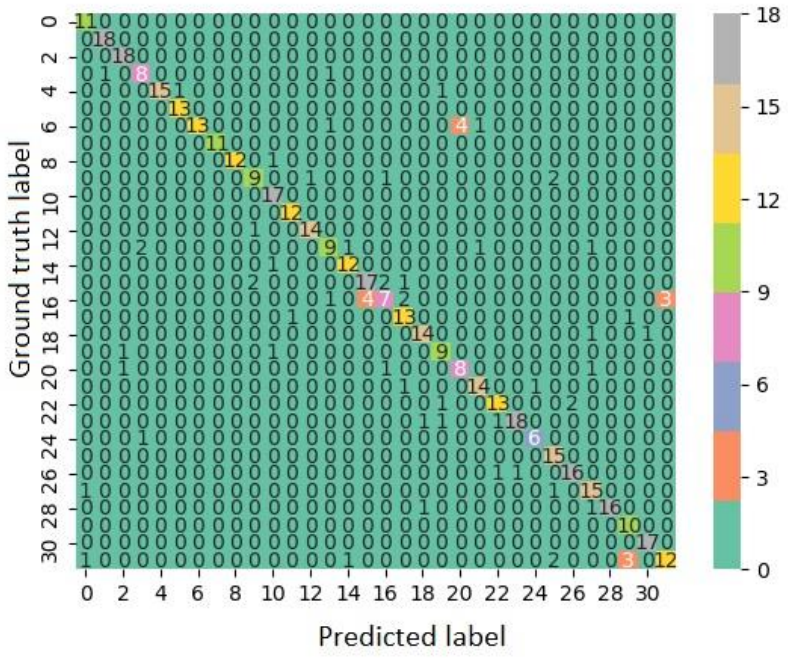


Figure 4.10 Confusion matrix obtained by using P-BPNN (Adam optimizer, Learning rate = 0.001, PReLU activation function with α = 0.1, Number of epochs = 200)

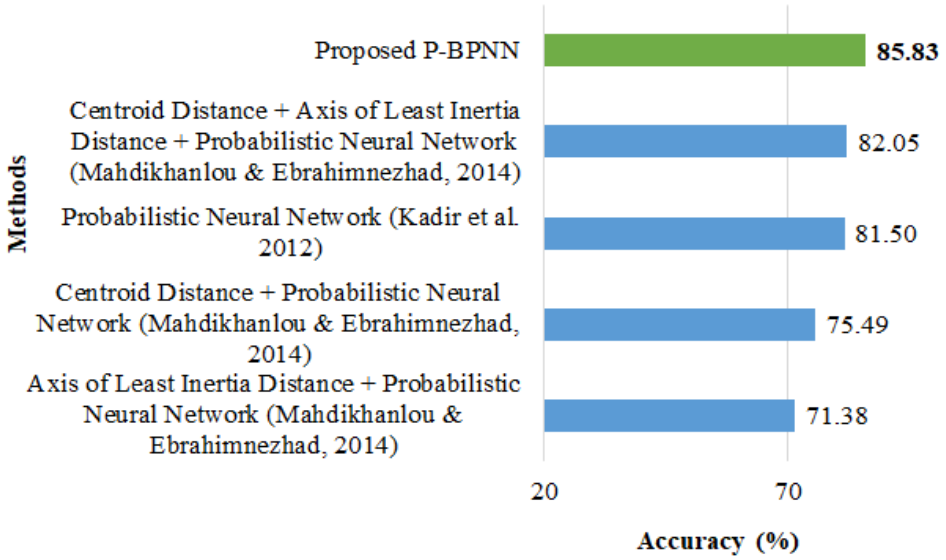


Figure 4.11 Comparison of existing literature with Proposed P-BPNN for Flavia dataset

Optimizer, Learning Rate = 0.001, PReLU Activation function with an α value of 0.1, and Number of epochs = 200.

4.4.2 Folio dataset

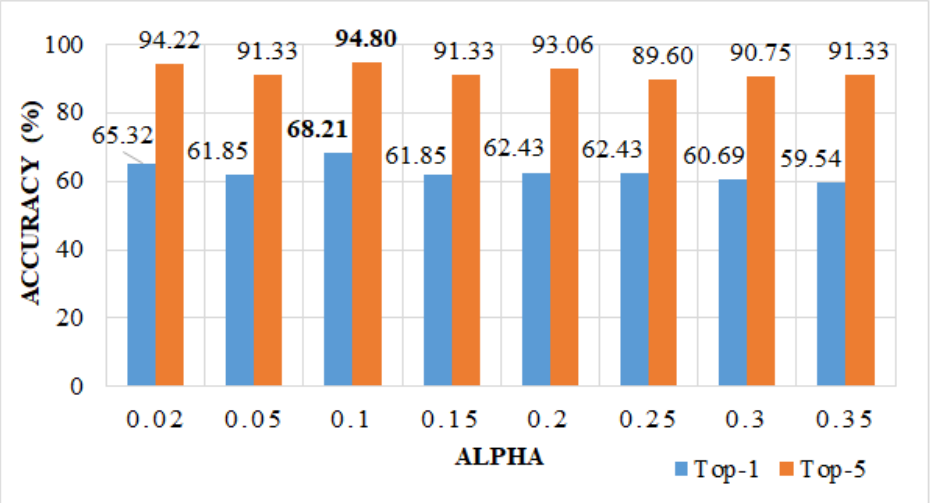
Similar to the Flavia dataset, the performance analysis of the proposed Neural Network architecture with Backpropagation algorithm is carried out on the Folio dataset. The optimization of different hyperparameters (optimizers, learning rate, activation function, α value for PReLU activation function, Epochs) is carried out, and the results are tabulated in Table 4.3. It is observed that the proposed Neural Network with Backpropagation algorithm performs better (*Top-1*: 65.32%, *Top-5*: 94.22%) when setting the hyperparameters as Adam optimizer, Learning rate = 0.001, PReLU Activation function with α of 0.02, Number of epochs = 50.

Table 4.3 Optimization of Hyperparameters for P-BPNN

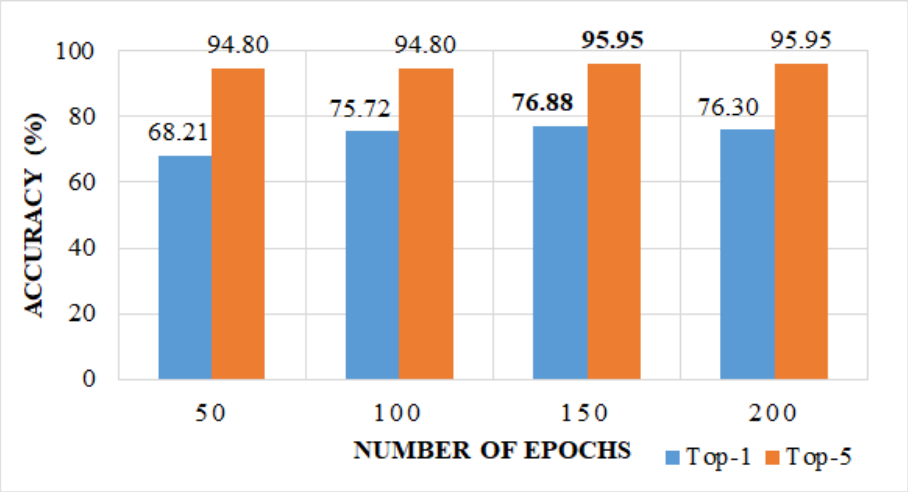
Optimizers	Accuracy (%)	
	Top-1	**Top-5**
SGD	32.37	74.57
Adam	**65.32**	**94.22**
Adamax	58.38	89.02
Adadelta	36.42	73.99
Learning Rate		
0.001	**65.32**	**94.22**
0.0001	53.18	86.71
0.00001	34.10	67.05
Activation function		
ReLU	61.27	92.49
PReLU	**65.32**	**94.22**
ELU	64.74	93.06
Leaky ReLU	54.34	83.82

Further analysis is carried out by varying the α value of the PReLU activation function. The results obtained for 50 epochs are shown in figure 4.12(a). It is observed that the proposed architecture performs better (*Top-1*: 68.21%, *Top-5*: 94.80%) when the PReLU activation function is set with an α value of 0.1. Later, the number of epochs is varied to determine the best optimization hyperparameters for recognizing the plant species. The results are shown in figure 4.12(b). It is observed that the improvement in the performance

(*Top-1*: 76.88%, *Top-5*: 95.95%) of the proposed system is achieved for 150 epochs.



(a)



(b)

Figure 4.12 Analysis of P-BPNN (a) Tuning Alpha (α) value for PReLU activation function (b) Variation in Number of epochs

The confusion matrix obtained for the proposed Neural Network with optimized hyperparameters (Adam optimizer, learning rate = 0.001, PReLU activation function with an α value of 0.1, Number of epochs = 150) is shown in figure 4.13. It is observed that the number of mispredictions/class is higher when compared to the Flavia dataset. This is due to the lesser number of

train and test images in the dataset. The Folio dataset contains only about 18 images/class. There are no related works on Neural Networks for Plant Species Recognition using the Folio dataset.

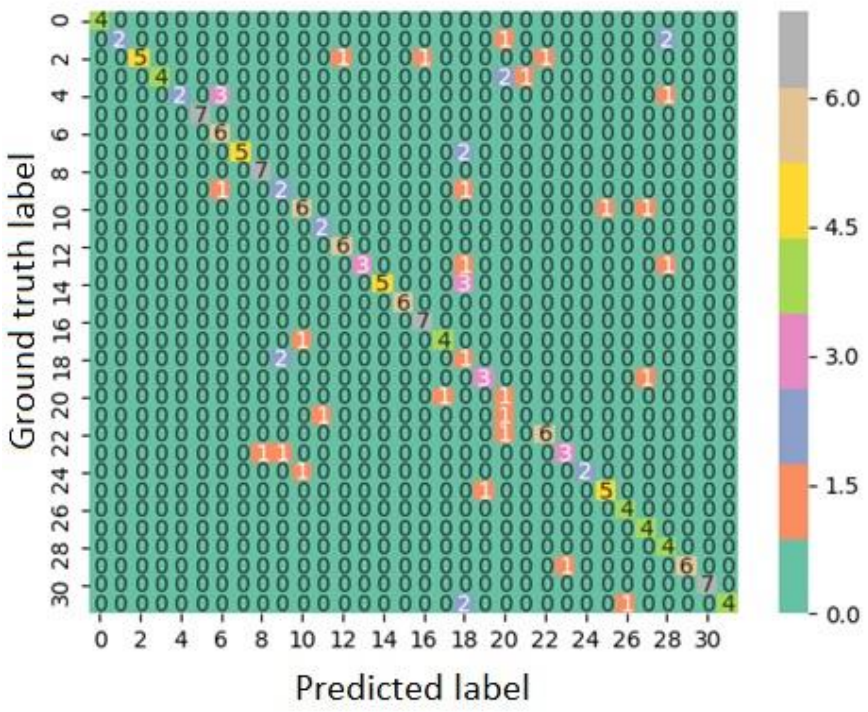


Figure 4.13 Confusion matrix obtained for P-BPNN (Adam optimizer, Learning rate = 0.001, PReLU activation function with α = 0.1, Number of epochs = 150)

4.4.3 Swedish leaf dataset

Similar to Flavia and Folio datasets, the performance analysis of the proposed Neural Network architecture with Backpropagation algorithm is carried out on the Swedish Leaf dataset. The optimization of different hyperparameters (optimizers, learning rate, activation function, α value for PReLU activation function, Epochs) is carried out, and the results are tabulated in Table 4.4. It is observed that the proposed Neural Network with Backpropagation algorithm performs better (*Top-1*: 92.60%, *Top-5*: 100%) by

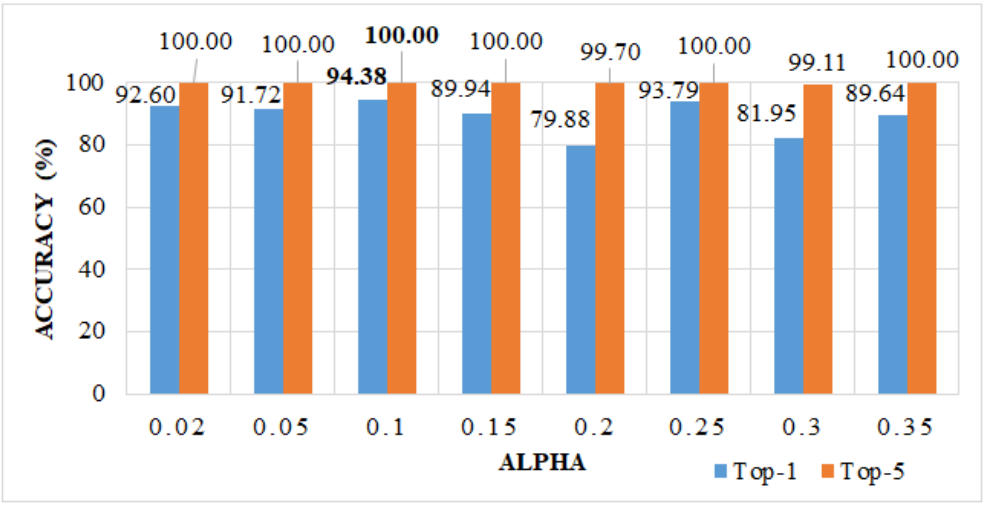
setting the hyperparameters such as Adam optimizer, Learning rate = 0.001, PReLU Activation function with α of 0.02, Number of epochs = 50).

Table 4.4 Optimization of Hyperparameters for P-BPNN

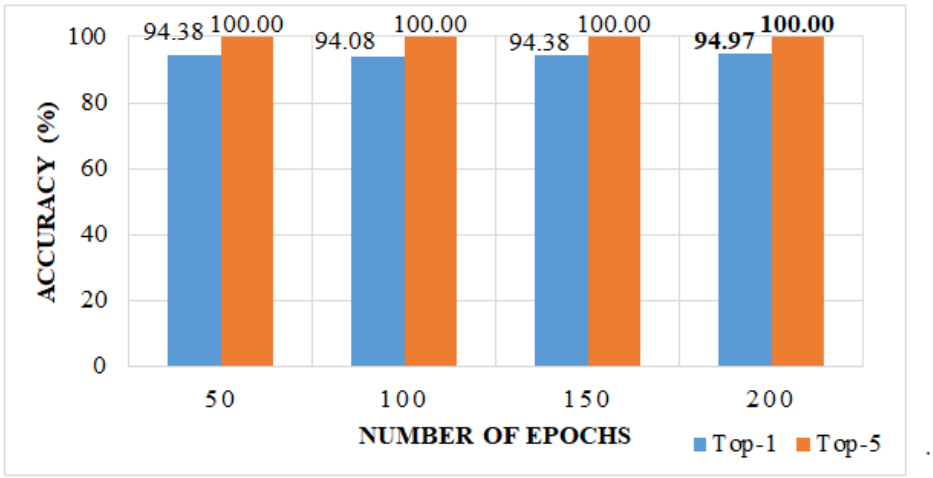
Optimizers	Accuracy (%)	
	Top-1	**Top-5**
SGD	76.33	99.41
Adam	**92.60**	**100**
Adamax	89.64	100
Adadelta	81.07	99.70
Learning Rate		
0.001	**92.60**	**100**
0.0001	89.35	100
0.00001	81.95	99.70
Activation function		
ReLU	92.01	100
PReLU	**92.60**	**100**
ELU	80.77	87.28
Leaky ReLU	79.88	100

Further analysis is carried out by varying the α value of the PReLU activation function. The results obtained at 50 epochs are shown in figure 4.14(a). It is observed that the proposed architecture performs better (*Top-1*: 94.38%, *Top-5*: 100%) when the PReLU activation function is set with an α value of 0.1. Later, the number of epochs is optimized to determine the best hyperparameters for the PSR system. The results are represented in figure 4.14(b). It is observed that the improvement in the performance (*Top-1*: 94.97%, *Top-5*: 100%) of the proposed system is achieved for 200 epochs.

The confusion matrix obtained for the proposed Neural Network with optimized hyperparameters (Adam optimizer, learning rate = 0.001, PReLU activation function with α value of 0.1, Number of epochs = 200) is shown in figure 4.15. It is observed that the number of mispredictions/class is greatly reduced compared to the Flavia and Folio dataset. This is due to the availability of more number of train and test images in the dataset.



(a)



(b)

Figure 4.14 Analysis of P-BPNN (a) Variation of Alpha (α) value in PReLU activation function (b) Variation in Number of epochs

To summarize, the proposed Neural Network architecture with Backpropagation algorithm performs better compared to other reported literatures, as shown in figure 4.16. The proposed architecture performs better with the hyperparameter settings: Adam Optimizer, Learning Rate = 0.001, PReLU Activation function with an α value of 0.1, Number of epochs = 200.

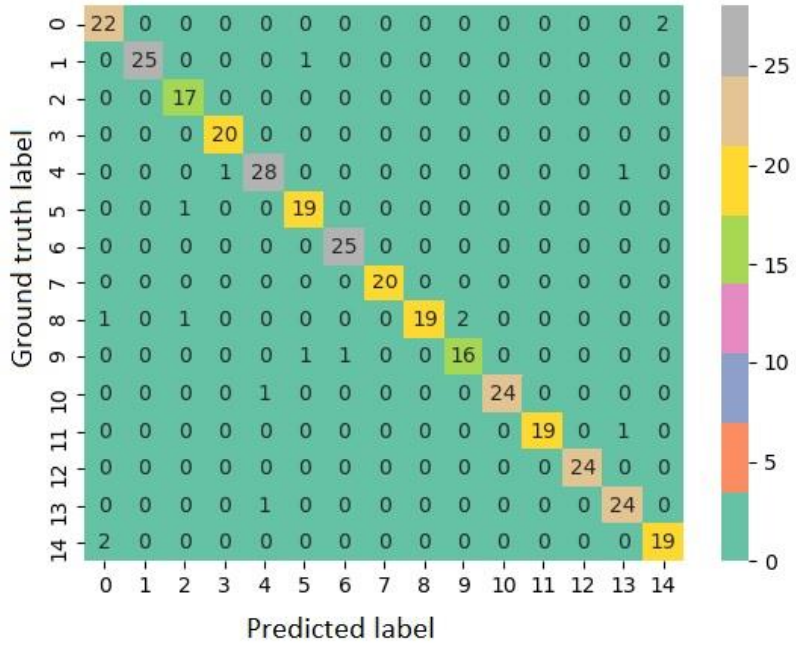


Figure 4.15 Confusion matrix of P-BPNN (Adam optimizer, Learning rate = 0.001, PReLU activation function with α = 0.1, Number of epochs = 200)

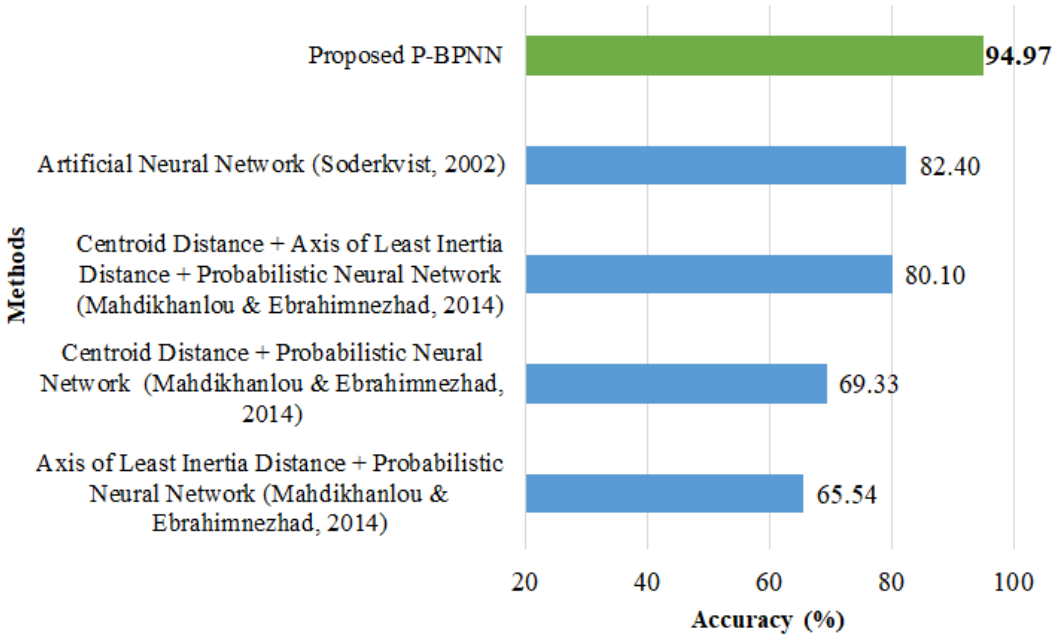


Figure 4.16 Comparison of Proposed P-BPNN with existing literature for Swedish leaf dataset

4.4.4 Leaf-12 dataset

Similar to Flavia, Folio, and Swedish leaf datasets, the performance analysis of the proposed Neural Network architecture with Backpropagation algorithm is carried out on custom developed Leaf-12 dataset. The dataset contains images captured under varying backgrounds, illumination conditions, and varying viewpoints or orientations of the object. The optimization of different hyperparameters (optimizers, learning rate, activation function, α value for PReLU activation function, Epochs) is carried out, and the results are tabulated in Table 4.5. It is observed that the proposed Neural Network with Backpropagation algorithm performs better (*Top-1*: 86.89%, *Top-5*: 99.39%) with the hyperparameter settings such as Adam optimizer, Learning rate = 0.001, PReLU Activation function with α of 0.02, and Number of epochs = 50.

Table 4.5 Optimization of Hyperparameters for P-BPNN

Optimizers	Accuracy (%)	
	Top-1	**Top-5**
SGD	80.82	98.87
Adam	**86.89**	**99.39**
Adamax	84.55	99.13
Adadelta	70.57	98.09
Learning Rate		
0.001	**86.89**	**99.39**
0.0001	82.64	98.96
0.00001	68.49	97.22
Activation Function		
ReLU	84.46	99.13
PReLU	**86.89**	**99.39**
ELU	76.30	98.35
Leaky ReLU	83.77	99.13

Further analysis is carried out by varying the α value of the PReLU activation function. The results obtained for 50 epochs are shown in figure 4.17(a). It is observed that the proposed architecture performs better (*Top-1*:

87.50%, *Top-5*: 99.41%) when the PReLU activation function is set with an α value of 0.1.

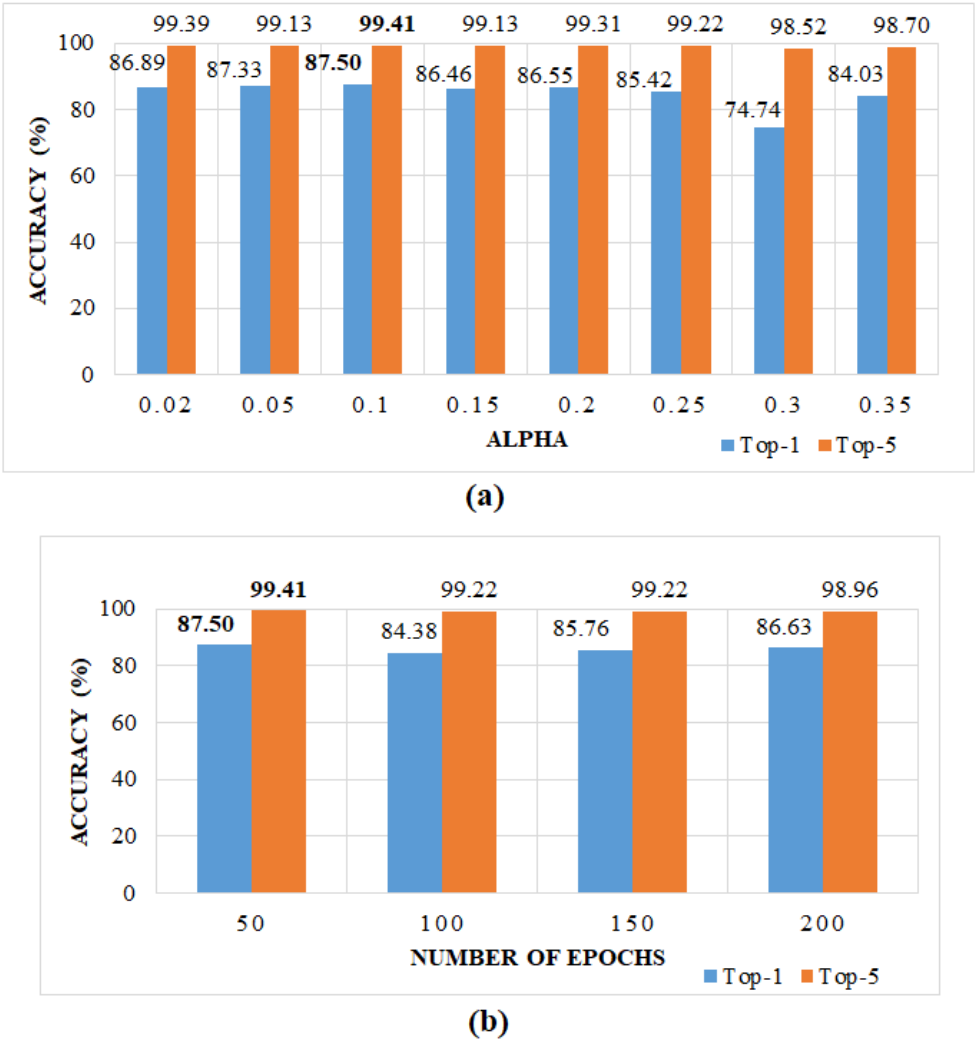


Figure 4.17 Analysis for P-BPNN (a) Variation of Alpha (α) value for PReLU activation function (b) Variation in Number of epochs

Later, the number of epochs is varied to determine the best optimization hyperparameters for plant species recognition. The results are represented in figure 4.17(b). It is observed that the improvement is not pronounced for higher values of epochs.

The confusion matrix obtained for the proposed Neural Network with optimized hyperparameters (Adam optimizer, Learning rate = 0.001, PReLU activation function with an α value of 0.1, Number of epochs = 50) is shown in figure 4.18. It is observed that the number of mispredictions/class is more pronounced compared to other datasets. This is due to the availability of train and test images with more variations (background, illumination, viewpoint, orientation, and scaling) in the dataset.

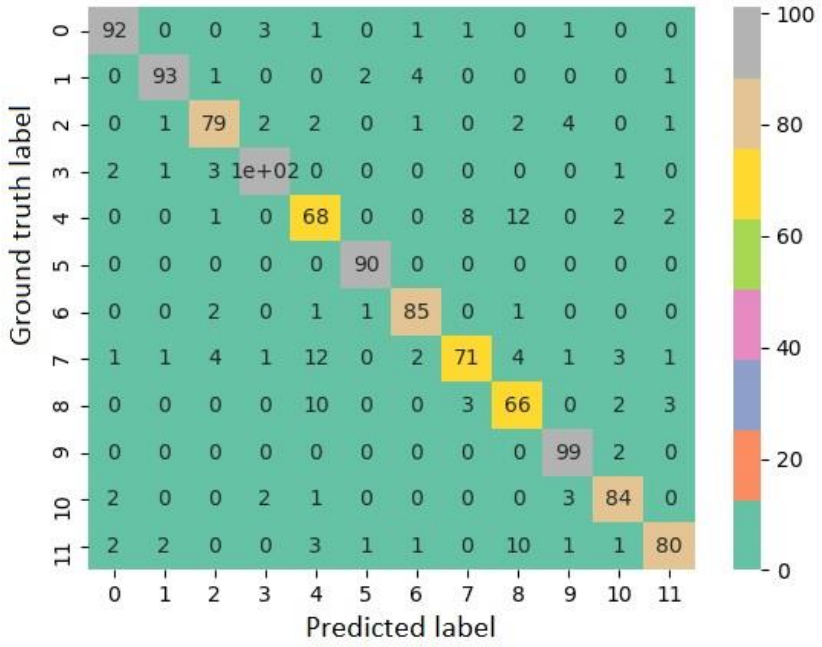


Figure 4.18 Confusion matrix obtained for P-BPNN (Adam optimizer, Learning rate = 0.001, PReLU activation function with α = 0.1, Number of epochs = 50)

4.5 SUMMARY

In this chapter, a simple two-layered Back-propagation Neural Network with PReLU activation function (P-BPNN) is investigated for Plant Species Recognition. Four datasets (Flavia, Folio, Swedish leaf, and Leaf-12) are considered in the evaluation of the proposed Neural Network. The images in the dataset are resized (maintaining the aspect ratio) and normalized (pixel

intensities), before feeding it as an input to the proposed Neural Network with Backpropagation algorithm. The hyperparameters (Optimizers, Learning Rate, Activation Function, Epochs) are optimized for the proposed architecture to attain better predictive accuracy.

The summary of results for the proposed architecture with the hyperparameter settings (Adam optimizer, Learning Rate = 0.001, PReLU activation function with an α value of 0.1) is shown in Table 4.6. The performance metrics such as *Top-1* accuracy, *Top-5* accuracy, Precision (*Pr*), Recall (*Re*), and F1-Score (*F1*) are tabulated. The results are summarized for the four datasets (Flavia, Folio, Swedish leaf, and Leaf-12).

Table 4.6 Performance Metrics of P-BPNN with Optimized Parameters

Dataset	Number of Epochs	Accuracy (%)		Precision, Pr	Recall, Re	F1-Score, F1
		Top-1	Top-5			
Flavia	200	85.83	98.75	0.86	0.86	0.85
Folio	150	76.88	95.95	0.85	0.77	0.78
Swedish leaf	200	94.97	100	0.95	0.95	0.95
Leaf-12	50	87.50	99.41	0.88	0.88	0.87

The performance of the PSR system could be further improved by using Deep Learning architectures. The methodology and the results obtained by using single and dual deep learning architectures are described in detail in the upcoming chapters (Chapter 5 - Single Deep learning architectures, Chapter 6 - Dual Deep learning architectures).

CHAPTER 5

PERFORMANCE ANALYSIS OF PLANT SPECIES RECOGNITION USING PRE-TRAINED DEEP LEARNING ARCHITECTURES

5.1 INTRODUCTION

In recent years, Deep Learning (DL) architecture has captured wider attention in Machine Learning by overcoming the limitations of the Backpropagation Neural Network. Convolutional Neural Network (CNN) (Goodfellow *et al.* 2016), Recurrent Neural Networks (RNN) (Goodfellow *et al.* 2016), Generative Adversarial Networks (GAN) (Goodfellow *et al.* 2014) and AutoEncoder (Baldi 2012) are some of the DL methods. CNNs are used in applications involving image detection and classification. RNNs are used in text processing or natural language processing. GANs are utilized in image generation and image editing. AutoEncoder is an unsupervised learning algorithm in Deep Learning.

Convolutional Neural Networks are also used in self-driving cars, medical applications (skin, breast or brain cancer detection), automatic image generation, automatic caption generation, and automatic text generation (Source: www.medium.com; www.machinelearningmastery.com). In this chapter, Convolutional Neural Networks are used in the context of Plant Species Recognition.

In the upcoming subsection, the methodology of CNN and transfer learning are discussed. Section 5.2 reports the literature survey on Plant Species Recognition systems using DL methods. Section 5.3 discusses the DL methods used in this dissertation for Plant Species Recognition. Section 5.4 provides the

experimental results obtained by using the DL methods for the four datasets. Section 5.5 gives a summary of this chapter.

5.1.1 Convolutional Neural Network

Convolutional Neural Network (CNN) (Goodfellow *et al.* 2016; Gu *et al.* 2018; Srinivas *et al.* 2016) consists of three layers, namely, Convolution layer, Pooling layer, and Fully connected layer. Several Deep Learning (DL) models such as VGG-16 (Simonyan & Zisserman 2014), VGG-19 (Simonyan & Zisserman 2014), Inception-V3 (Szegedy *et al.* 2016), Inception ResNet-V2 (Szegedy *et al.* 2016), Xception (Chollet 2017), MobileNet (Howard *et al.* 2017), DenseNet-121 (Huang *et al.* 2017), DenseNet-169 (Huang *et al.* 2017), and DenseNet-201 (Huang *et al.* 2017) utilize the CNN methodology.

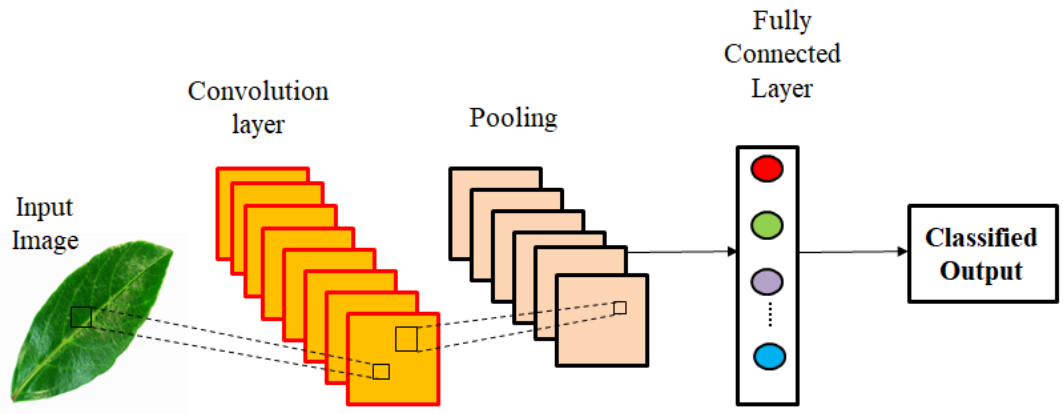


Figure 5.1 Convolutional Neural Network

Convolutional Neural Network (CNN) is a supervised learning method. It utilizes the Backpropagation algorithm to update the weights. CNN performs both feature extraction and classification. This characteristic of CNN makes it more suitable for real-time scenarios. Figure 5.1 shows the block diagram representation of CNN.

5.1.1.1 Convolution Layer

Convolution layer utilizes the convolution process for extracting the features from the image. Convolution is an image processing operation and is represented in Equation (5.1). In the Convolution process, the filter slides over an input image, as shown in Figure 5.2 (Goodfellow *et al.* 2016). The filter contains weight values. Initially, the weights are assigned in a random fashion. In simple terms, convolution is the process of multiplication and addition of inputs and weights. The filter slides over the entire image to perform the convolution process, resulting in a matrix named as a feature map.

$$Conv(i,j) = \sum_m \sum_n f(m,n).K(i-m, j-n) \quad (5.1)$$

Here, *Conv(i, j)* represents the Convolution Matrix

f (m,n) denotes the image

K is the filter or weight matrix or kernel

Stride (*s*) is a hyperparameter, indicating the number of pixel position the filter slides over the image. Usually, the stride value is set to be one or two. Padding (*pd*) is an optional parameter to preserve the spatial information of the image. Commonly, zero padding is used. If the image dimension is represented by *m* x *n*, the output dimension (*O*) (Dumoulin & Visin 2016) of a convolution layer is calculated as follows,

$$O = \left[\frac{m+2pd-F}{s}\right] + 1 \quad (5.2)$$

where *m* represents the width of the image; *F* is the size of the filter; *s* indicates the stride value, and *pd* denotes padding. After the convolution operation, the feature maps are dimensionally reduced by the pooling method. The above specified Equation (5.2) can be rewritten with respect to height and width of the image as follows.

$$Output\ height = \frac{Input_n + pd_{n_t} + pd_{n_b} - F_n}{s_n} + 1 \qquad (5.3)$$

Where, $Input_n$ is the input height of the image; pd_{n_t} is padding height top; pd_{n_b} is padding height bottom. F_n is filter height and s_n is stride height.

$$Output\ width = \frac{Input_m + pd_{m_r} + pd_{m_l} - F_m}{s_m} + 1 \qquad (5.4)$$

Where, $Input_m$ is the input width of the image, pd_{m_r} is padding width right; pd_{m_l} is padding width left; F_m is Filter width; and s_m stride width. (Source: https://iq.opengenus.org/output-size-of-convolution/)

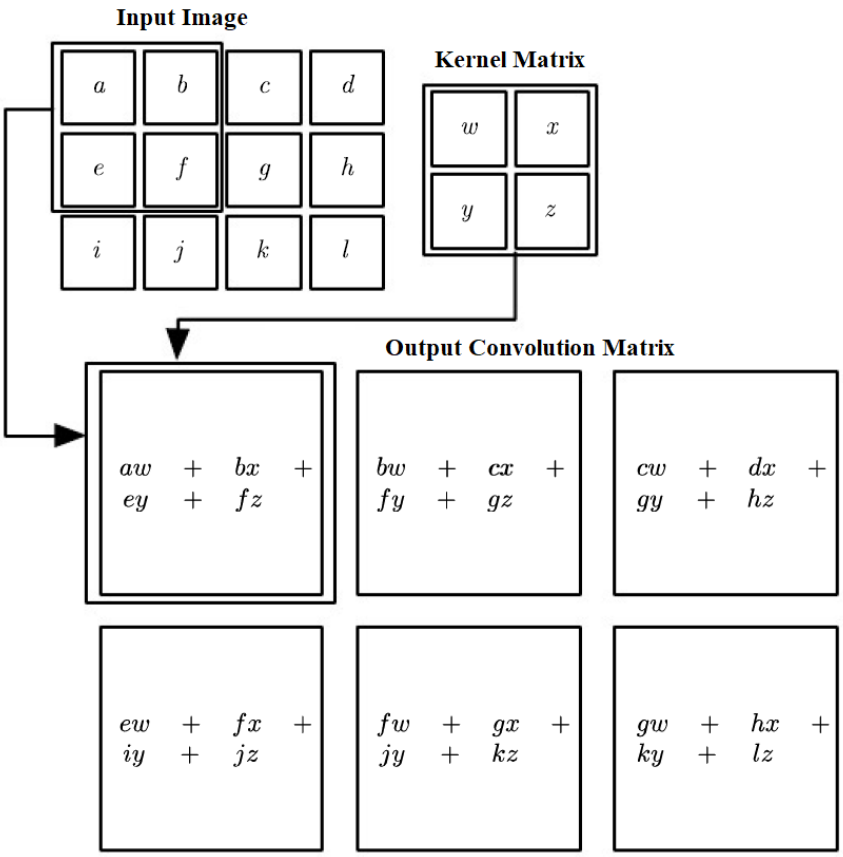


Figure 5.2 Convolution operation (Source: Goodfellow *et al.* 2016)

5.1.1.2 Pooling Layer

The pooling layer reduces the dimension of the feature maps (Gogul & Kumar 2017). The dimensionally reduced feature map is forwarded to the

next layer for further processing. Reducing the dimension of feature maps result in a computationally efficient system (Goodfellow 2016).

Usually, a 2x2 window or filter is used for the pooling operation. There are two types of pooling methods, namely, Average and Max pooling. Figure 5.3 represents an example of pooling operations. Average pooling determines the average value of the chosen window in the feature map. Max pooling outputs the maximum value among the chosen window. This results in the selection of highly essential features. The dimensionally reduced feature map is given as an input to a fully connected layer.

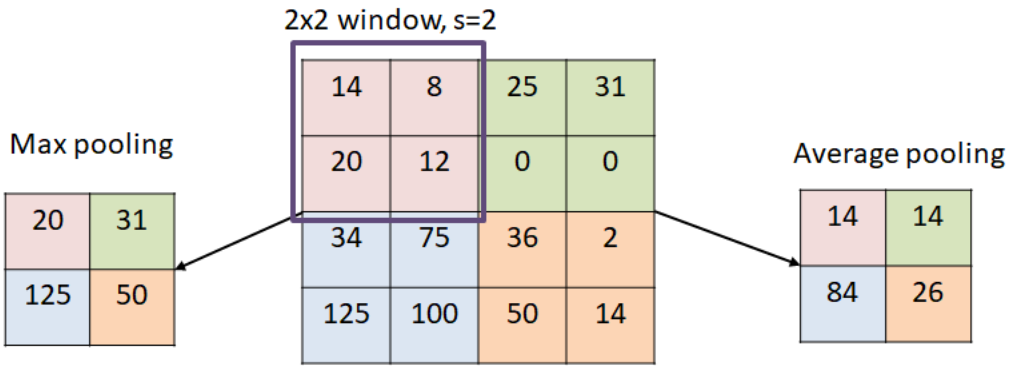


Figure 5.3 Max pooling and average pooling operation on feature map using a window size of 2x2 and stride value (*s*) of 2

5.1.1.3 Fully Connected Layers

A Fully Connected Layer (FCL) is used for predicting the output class (Gogul & Kumar 2017). In CNN, two or three fully connected layers are used. The final FCL uses a 'Softmax' activation function for classification. CNN utilizes the Backpropagation algorithm. In this layer, the errors (ground truth-predicted value) are computed using the loss function (Categorical Cross-Entropy). The weights are updated in a backpropagated manner to minimize the loss function. Weight updation is performed by different optimizers such as

Adam, Stochastic Gradient Descent (SGD), Adamax, and Adadelta. It results in the improvement of accuracy in prediction.

5.1.2 Transfer Learning

Transfer learning (Pan & Yang 2009; Weiss *et al.* 2016) is a method in machine learning. It utilizes the knowledge obtained by performing one particular task and apply it to another related process. ImageNet weights are used in the pre-trained CNN models. Kornblith *et al.* (2019) proved that the usage of ImageNet weights resulted in faster convergence. These weights result in a computationally efficient system (low training time). ImageNet weights are used as initial weights for all the Deep Learning architectures, considered in this chapter as well as Chapter 6.

The pre-trained models are utilized in two ways (Srinivas *et al.* 2016). The first approach is to use the pre-trained CNN model as a feature extractor. The final fully connected layer (FCL) is removed to use it as a feature extractor. The extracted features are then classified by using Machine Learning classifiers. The second approach is by fine-tuning the pre-trained CNN models. It is performed by retraining the final fully connected layer (FCL). The final fully connected layer consists of output neurons equal to the number of output classes. The softmax activation function is used in the final FCL.

Figure 5.4 depicts the fine-tuning of the pre-trained CNN model. In the Fine-tuning method, the final fully connected layer is modified, leaving the convolution base to be unchanged, as suggested by Chollet (2017). The convolution base consists of more generic features in comparison to FCL. FCLs are problem-specific. Hence, the final fully connected layer is not recommended for new applications, which varies drastically from the original applications. In this dissertation, Chollet's approach (2017) is followed to obtain the results for Deep Learning architectures.

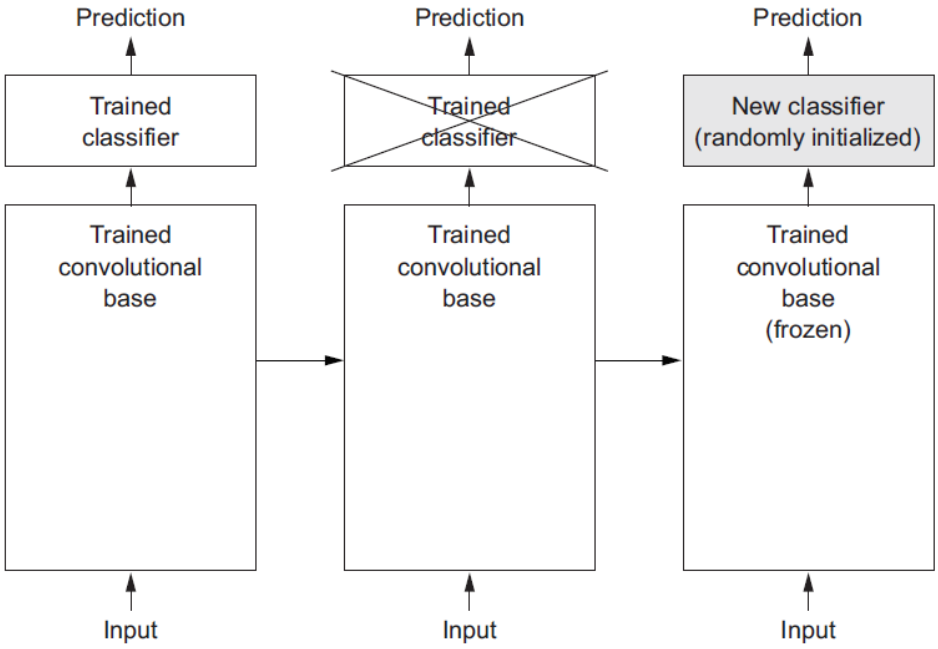


Figure 5.4 Fine-tuning of Pre-trained CNN model (Source: Chollet 2017)

5.2 SURVEY ON DEEP LEARNING ARCHITECTURES FOR PLANT SPECIES RECOGNITION

An extensive survey on deep learning-based plant species recognition methods is presented in this section. This survey focuses on two major subsections, namely, custom-developed CNN models and Transfer

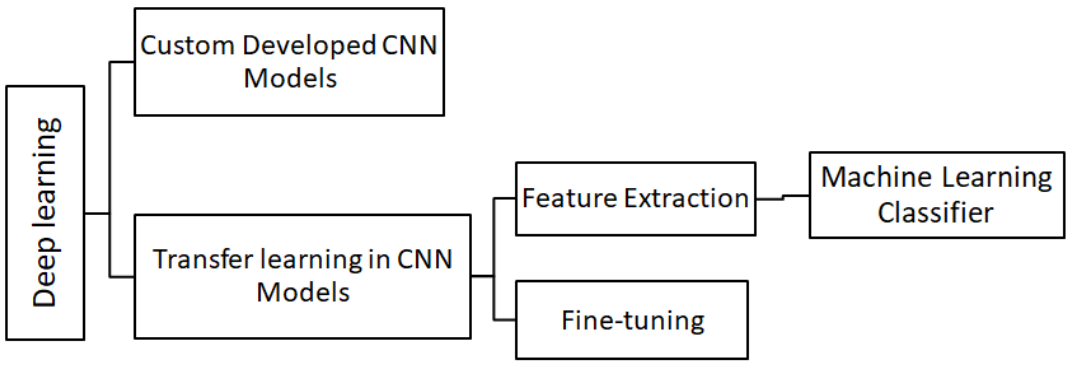


Figure 5.5 Hierarchy for Plant Recognition using Deep learning

learning of CNN models, as shown in figure 5.5. Subsection 5.2.1 discusses the usage of custom-developed CNN models in plant species recognition. Subsection 5.2.2 discusses the usage of Transfer Learning in CNN models for plant species recognition.

5.2.1 Custom Developed CNN Models

Younis *et al.* (2018) proposed a modified ResNet CNN architecture containing 50 layers for plant species recognition. It utilizes the herbarium specimens for classification. The authors utilized an Exponential Linear Unit (ELU) instead of the ReLU activation function. A custom dimension of 225x225x3 is used for the scanned images. The vascular herbarium images (1000 species) are obtained from Muséum National d'Histoire Naturelle (MNHN). The model is trained with Adam optimizer using Nesterov Momentum, learning rate as 1e-4, batch size of 60, and 20,000 steps. For trait recognition, a plain ResNet CNN architecture's last layer is trained with a sigmoid activation function.

Carranza-Rojas *et al.* (2017) proposed an extended version of GoogleNet CNN architecture. This model utilizes the Inception modules. The difference between the author's model and GoogleNet CNN architecture is in the usage of the activation function. The author's model utilizes the PReLU activation function instead of the ReLU activation function. Also, the model utilizes the Batch Normalization for Inception modules. This model is trained with a learning rate of 0.0075 and a batch size of 16. For training, random weight initialization and ImageNet weights are used. The datasets considered are Costa Rica, Herbarium255, PlantCLEF2015, and Herbarium1K. Also, experiments are carried out on cross-dataset transfer learning. It is observed that the fine-tuned model using ImageNet weights or a combination of weights (ImageNet + Herbarium-255K and ImageNet + Herbarium-255K) improved the accuracies.

Sun *et al.* (2017) constructed a 26-layer ResNet (Residual Network) CNN architecture. This ResNet CNN model has about eight residual blocks. ResNet-26 CNN model surpassed with improved results compared to 18, 34, and 50-layered ResNet CNN models. It is trained using Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.001, a decay rate of 10^{-6} and momentum of 0.9. This model is tested using a real-time dataset, 'BJFU100'. The dataset consists of 10000 images from 100 ornamental plant species collected at the Beijing Forestry University campus. This method is also evaluated using a standard leaf dataset called 'Flavia'.

LeafNet CNN architecture (Barre *et al.* 2017) is designed for automated identification of plant species. The method is evaluated using Leafsnap, Foliage, and Flavia datasets. Label preserved data augmented transformations are carried out to reduce the overfitting. The LeafNet CNN architecture consists of five dimension reduction modules, one convolution layer, one max pooling layer, and three fully connected layers (2048, 2048, and 185 neurons in each FCL). In each reduction module, the max-pooling layer is followed by two convolution layers. Leafsnap dataset is trained using Momentum of 0.9, mini-batch size of 10 training images, a learning rate of 0.001, and gamma of 0.1. The number of iterations varied for each dataset, and it is about 2/3 size of the dataset. LeafNet CNN architecture consumed about 32 hours for training 2,70,000 images, using GPU (NVIDIA GTX 960). The accuracies obtained for LeafSnap, Foliage, and Flavia datasets are 86.3%, 95.8%, and 97.9%, respectively.

Zhang *et al.* (2015) proposed a seven-layered ConvNet (CNN) architecture for leaf recognition. This ConvNet CNN architecture consists of three convolution layers with max-pooling, local response normalization (LRN), dropout, PReLU activation function, and one FCL. The method is

applied to augmented (rotation, translation, scaling, contrast, sharpening) Flavia dataset. It resulted in an accuracy of about 94.6%.

Atabay (2016) proposed a CNN model with ELU activation function. The CNN architecture comprises of three convolutions with 5x5 filters. Each convolution layer is followed by the ELU activation function and 5x5 max-pooling layers. The fourth layer is a 1x1 convolution layer with a Softmax activation function. Green channel of Flavia and Swedish leaf datasets are passed through the CNN model and trained using Stochastic Gradient Descent (SGD) with Momentum optimizer. The model produced better accuracies (Flavia-97.24% and Swedish leaf-99.11%) using the ELU activation function compared to the ReLU activation function.

Yalcin & Razavi (2016) employed CNN for categorizing the plant species from smart agro-stations. Illumination variations and deblurring effects are minimized by preprocessing the images. The CNN architecture is similar to the AlexNet CNN architecture except for the variation in the pooling layer (window size of 2x2, whereas AlexNet CNN uses a 3x3). The number of plant species considered in the studies is 16. The results of the proposed method are compared with the conventional method. In the conventional method, the features are extracted by using the Local Binary Pattern and GIST technique. Later, the extracted features are classified using Support Vector Machine (SVM) classifier. The CNN outperformed the other method with an accuracy of 97.47%.

5.2.2 Transfer learning in CNN models

5.2.2.1 CNN as Feature Extractor

Kaya *et al.* (2019) employed the types of transfer learning techniques and a custom created CNN model for plant species recognition. The

custom-developed CNN model consists of three convolution layers and a FCL. The second approach is the fine-tuning of the custom-developed CNN model for the four datasets. Fine-tuning of AlexNet and VGG-16 CNN architectures are also reported by the authors. The fourth and fifth approaches are deep feature extraction process utilizing the AlexNet and VGG-16, respectively. In the fourth approach, the last FCL is replaced with linear Support Vector Machine (SVM) and Linear Discriminant Analysis (LDA). In the fifth approach, the last FCL is replaced with Recurrent Neural Network (RNN). The aforementioned approaches are evaluated using UCI leaf, Plantvillage, Flavia, and Swedish leaf datasets. The results showed that the transfer learning approaches performed much better than end-to-end CNN architectures.

Tan *et al.* (2018) created a CNN model termed as D-Leaf for feature extraction. Five machine-learning classifiers, namely, Support Vector Machine (SVM), Artificial Neural Network (ANN), K-Nearest Neighbor (K-NN), Naïve Bayes (NB) and Convolutional Neural Network (CNN) are used for classification. Similarly, features extracted from pre-trained and fine-tuned AlexNet CNN architectures are tested with the same classifiers. The authors also reported the results for the conventional approach. In the conventional approach, the features are extracted from Sobel segmented images and classified using ANN. It is inferred that the performance of the fine-tuned AlexNet CNN model is superior to D-Leaf CNN architecture and conventional approach. The methods are evaluated using the self-collected dataset, Malayakew, Flavia, and Swedish leaf datasets.

Zhu *et al.* (2018) proposed a VGG-19 CNN architecture with linear SVM for plant species recognition. Sixteen convolution layers and three FCLs (4096 neurons, 4096 neurons, and 1000 neurons) are used. The extracted features are progressed to the linear SVM. Two types of features (Feature-1 and Feature-2) are extracted. Feature-1 contained the results of FCLs without

activation functions (ReLU and Softmax). Feature-2 is formed from the results of FCLs with activation functions. The method is evaluated with sub-datasets (leaf, flower, fruit, branches, and stem) of the LifeCLEF2015 dataset. Feature-2 based classification resulted in higher accuracy compared to Feature-1 based classification.

Zhu *et al.* (2018) proposed a Faster-RCNN model with Inception-V2 as a feature extractor for plant species recognition. This method tends to work better under unconstrained environments. Resizing of images is carried out, before sending it to the CNN model. The output of the CNN model is passed on to the Softmax activation function and bounding box regressor to predict the labels.

Zhang & Zhang (2017) proposed a Deep Convolutional Neural Network (DCNN) for plant species recognition to address the poor convergence and generalization problems. DCNN consists of nine layers, which includes one input layer, five convolution layers, and 3 FCLs (4096, 4096, output neurons vary with the number of classes). The images in the Swedish Leaf Dataset are augmented using the horizontal reflection method. DCNN model utilizes the AlexNet CNN architecture with training parameters such as initial learning rate of 10^{-5} and reduced by /10, for every 20,000 successive iterations. DCNN is used for feature extraction, and the linear Support Vector Machine (SVM) is used for classification. It resulted in better prediction using a five-fold cross-validation scheme. The computation time for training samples is about 15 hours.

Beikmohammadi & Faez (2018) utilized a pre-trained deep learning model to extract the leaf information from RGB input images. The pre-trained DL model (MobileNet) is used as a feature extractor. The extracted features are classified using Logistic Regression (LR) classifier. The model is evaluated with Flavia and LeafSnap datasets. It is observed that the transfer learning

method is best suited for plant species recognition. The results are obtained with low computational resources and time.

5.2.2.2 Fine-tuned CNN Models

Krause *et al.* (2019) classified the plant species, addressing the challenges such as inter-species similarity and more number of plants. A multi-scale approach is used to separate the regions of interest resulting in multi-scale patches. Various CNN architectures such as ResNet-18, ResNet-34, ResNet-50, Xception, Inception-V3, and Inception ResNet-V2 are trained using the multi-scale patches. An accuracy of 89.21% is obtained for the UHManoa100 dataset using Inception ResNet-V2 CNN architecture.

Bodhwani *et al.* (2019) presented Residual Network for the identification of plant species. The residual network contained 50 layers with skip connections. Also, data augmentation is performed to improve the accuracy. The method is trained using Adam optimizer with a learning rate of 0.001, batch size of 120, and the number of epochs as 41. ResNet-50 CNN architecture is evaluated using the LeafSnap dataset. It resulted in an accuracy of 93.09%.

Nguyen *et al.* (2017) evaluated the Conventional and Deep Learning approaches for plant species recognition. ImageCLEF 2015 dataset is used to evaluate the method. In the conventional approach, Kernel Descriptor (KDES) with multi-class SVM is used. For deep learning methodology, GoogleNet CNN architecture is trained with leaf and flower images. The parameters used for training are learning rate of 0.001, batch size of 5, and 50,000 number of iterations. For plant species recognition, the individual accuracy values obtained for leaves and flowers are fused to produce the final predictions. It resulted in a Rank-1 accuracy of 66.46%.

Pawara *et al.* (2017) identified plant species using both Traditional and Deep Learning methods. Three datasets (AgrilPlant, LeafSnap, and Folio datasets) are used in the studies. AgrilPlant is a self-collected dataset. It consists of 10 agricultural plants (300 images per class). Deep learning models such as AlexNet and GoogleNet CNN architectures are analyzed using two approaches, such as scratch and fine-tuned method. The traditional methods such as Histogram of Oriented Gradient (HOG) and Bag of visual Words (BoW) are used as feature extractors. The study showed that the performance of CNN models outperformed the conventional methods.

Dias & Borges (2016) collected fifty plant species from Brazil to form a Plantas dataset. The fine-tuned CNN architectures, namely, AlexNet, CaffeNet, GoogLeNet, Inception, and ResNet, are evaluated with the Plantas dataset. The CNN architectures are also tested by varying the activation functions (ReLU, PReLU, and ELU). For validating the results, the Plantas dataset is evaluated using Encoding methods such as Bag of visual Words (BoW), Fisher Vectors (FV), and Vector of Locally Aggregated Descriptors (VLAD). The methods specified above are used as feature extractors. Then, it is classified using SVM classifier. CNNs are trained for 30 epochs with an initial learning rate of 0.01, reduced by 10 for every 10 epochs. Other parameters are set as 0.0005 (weight decay) and 0.9 (momentum). Batch size for each CNN architecture is varied due to the memory constraints. Data augmentation is performed by random cropping and mirroring methods. It is observed that the fine-tuned GoogleNet CNN architecture with the PReLU activation function resulted in higher accuracy (94.5%) compared to other methods.

5.2.3 Limitations of reported literature on Deep Learning Architectures for Plant Species Recognition

Table 5.1 summarizes the usage of deep learning architectures in the context of plant species recognition. An extensive literature review is carried out to identify the research gaps in deep learning-based PSR system.

In most of the reported literatures, standard input image size (For example:- VGG-16: 224x224x3, Inception-V3: 299x299x3, DenseNet-121: 224x224x3 etc.) as in the original architecture is used for evaluating the performance of each of the CNN architectures. It is identified that the Transfer Learning method (Fine-tuning or CNN as feature extractor) performed better in comparison to Traditional methods and Neural Networks. The accuracy is improved using the data augmentation process (Barre *et al.* 2017; Atabay 2016). Fine-tuned CNN architectures resulted in higher accuracy with increased computation time. By utilizing the pre-trained CNN architecture as a feature extractor, a comparable accuracy (with fine-tuned CNN architectures) with lower computation time is achievable.

Table 5.1 Summary of reported Deep Learning Architectures for Plant Species Recognition

S. No.	CNN models	Transfer learning		Author & Year
		CNN as Feature Extractor	Fine-tuned CNNs	
1.	End-to-end CNN	AlexNet+ Support Vector Machine (SVM), AlexNet+ Linear Discriminant Analysis (LDA), VGG-16 + SVM, VGG-16 + LDA	AlexNet, VGG-16 AlexNet + Recurrent Neural Network (RNN), VGG-16 + RNN	Kaya *et al.* (2019)

Table 5.1 (Continued)

S. No.	CNN models	Transfer learning		Author & Year
		CNN as Feature Extractor	Fine-tuned CNNs	
2.	-	-	ResNet-18, ResNet-34, ResNet-50, Xception, Inception-v3, Inception ResNet-V2	Krause *et al.* (2019)
3.	-	-	ResNet-50	Bodhwani *et al.* (2019)
4.	ResNet-50 with Exponential Linear Unit (ELU)	-	-	Younis *et al.* (2018)
5.	-	MobileNet+ Logistic Regression (LR)	-	Beikmohammadi & Faez (2018)
6.	-	D-Leaf + SVM, D-Leaf + Artificial Neural Network (ANN), D-Leaf + K-Nearest Neighbor (KNN), D-Leaf + Naïve Bayes (NB)	D-Leaf	Tan *et al.* (2018)
7.	-	VGG 19 + linear SVM	-	Zhu *et al.* (2018)
8.	-	Inception-V2	-	Zhu *et al.* (2018)
9.	GoogleNet with PReLU	-	-	Carranza-Roja *et al.* (2017)

Table 5.1 (Continued)

S. No.	CNN models	Transfer learning		Author & Year
		CNN as Feature Extractor	Fine-tuned CNNs	
10.	26-layered ResNet	-	-	Sun *et al.* (2017)
11.	LeafNet architecture	-	-	Barre *et al.* (2017)
12.	-	Deep Convolutional Neural Network (DCNN-AlexNet) + linear SVM	-	Zhang & Zhang (2017)
13.	-	-	CNN (GoogleNet)	Nguyen *et al.* (2017)
14.	-	-	AlexNet and GoogleNet	Pawara *et al.* (2017)
15.	Seven-layered ConvNet	-	-	Zhang *et al.* (2015)
16.	CNN with ELU	-	-	Atabay (2016)
17.	CNN	-	-	Yalcin & Razavi (2016)
18.	-	-	AlexNet, CaffeNet, GoogLeNet, Inception, and ResNet	Dias & Borges (2016)

Hence, in this chapter, pre-trained CNN architectures as a feature extractor and fine-tuned CNNs are studied. The results are obtained without data augmentation process as the process increases computational time. Also, GAP layer requires more computational time. This chapter results are recorded

without Global Average Pooling (GAP) layer for models possessing the GAP layer.

5.3 METHODOLOGY

The methodology for plant species recognition is represented in Figure 5.6. Plant images from the dataset are resized (100x100) and forwarded to the Deep Learning (*M*) architecture model. Two approaches are followed in Deep Learning architectures, i.e., with and without fully connected layers (FCL). In both the approaches, the pre-trained weights from the ImageNet dataset are used. The global average pooling (GAP) and FCLs are removed from the CNN models (*M*), irrespective of the two approaches.

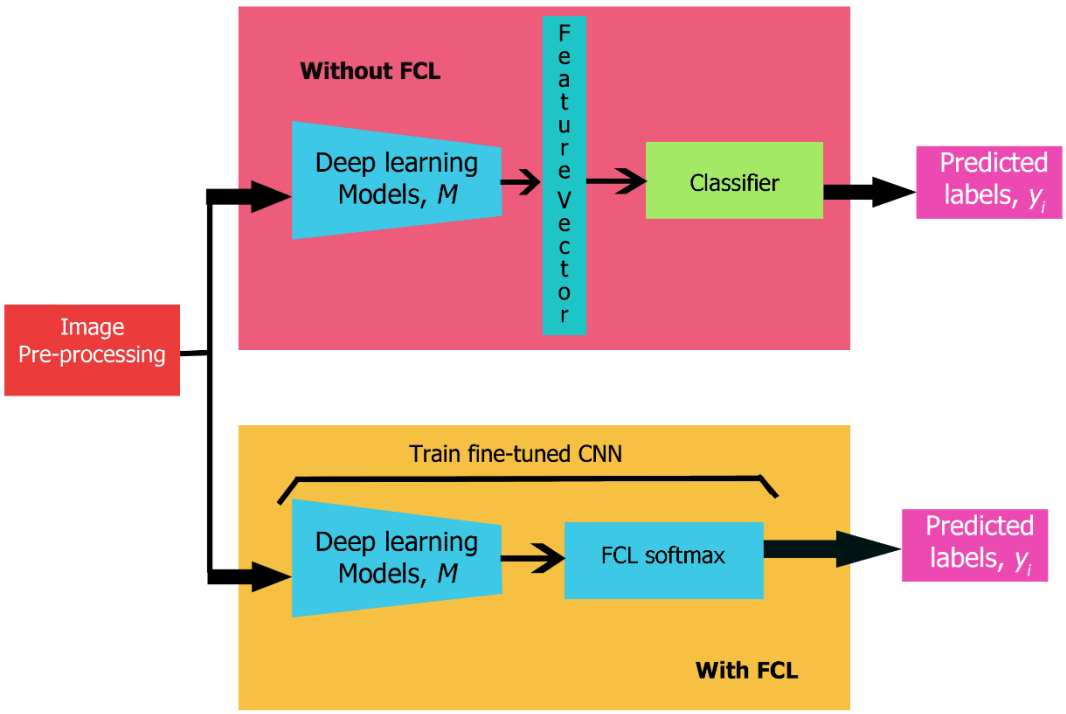


Figure 5.6 System Diagram for Plant Species Recognition using Pre-trained Deep Learning Architectures

In the first approach, several pre-trained DL models (1 to *M*) are used as feature extractors. It is described in section 5.3.1. The extracted features form a 1-D feature vector. The size of the feature vector for each DL model (*M*)

is shown in Table 5.2. To the feature vectors, conventional machine learning methods such as Linear Discriminant Analysis (LDA), Logistic Regression (LR), Naïve Bayes (NB), K-Nearest Neighbor (K-NN), Classification and Regression Tree (CART), Random Forest Classifier (RFC), Bagging Classifier (BC), Multi-Layer Perceptron (MLP) and Support Vector Classifier (SVM) are applied to predict the labels.

In the second approach, a single FCL with Softmax activation function is added to the DL model (*M*). This approach is trained for 50 epochs. It is explained in subsection 5.3.2.

Table 5.2 Size of Feature Vector for DL Architectures

CNN Models, *M*	Number of GAP and FCL	Feature Vector size
VGG-16	3 FCLs	4608
VGG-19	3 FCLs	4608
Inception-V3	1 GAP, 1 FCL	2048
Resnet-50	1 GAP, 1 FCL	32768
Inception Resnet-V2	1 GAP, 1 FCL	1536
Xception	1 GAP, 1 FCL	18432
Densenet-121	1 GAP, 1 FCL	9216
Densenet-169	1 GAP, 1 FCL	14976
Densenet-201	1 GAP, 1 FCL	17280

5.3.1 Approach-1

Images are resized to 100x100x3. The features are extracted from the *M* number of CNN architectures, without using either Global Average Pooling (GAP) or FCL. Table 5.2 lists the number of GAPs and FCLs used by each of the CNN models, *M*. Algorithm 5.1 describes the methodology applied to extract the features using Deep Learning architectures. Also, the algorithm provides details on the classification process.

Algorithm 5.1: CNN for feature extraction

Input: Image, I_N; CNN Model, M; Layer, L; Classifier, Cl; ImageNet weights, IW and biases, IB

Output: Feature Vector, X; Predicted label, y_i

Step 1: Read the N number of images (I_N)

Step 2: Resize the I_N images to 100x100x3

Step 3: Pass the I_N images through the convolution (Equation 5.1) and pooling layers (Equation 5.2) of the CNN model, M using IW and IB.

Step 4: Obtain the feature vector, X from the L^{th} layer of the CNN model, M.

$$X = \sum_{k=1}^{N} pred(M_j), j = 1, \dots, M$$

Step 5: Save the features, X in hdf5 file format

Step 6: Apply the classifier, Cl to the features, X to predict the label, y_i

There are M CNN Models used for the analysis. Nine machine learning classifiers (Cl_1 to Cl_9) namely, Linear Discriminant Analysis (LDA), Logistic Regression (LR), Classification and Regression Tree (CART), Naïve Bayes (NB), K-Nearest Neighbor (K-NN), Random Forest Classifier (RFC), Bagging Classifier (BC), Multi-Layer Perceptron (MLP) and Support Vector Machine (SVM), are applied.

Since the FCLs are not used, the images are processed only once (Chollet 2017). Hence, the weights and biases are not updated in this approach. The ImageNet weights are the initial weights used for all the M CNN architectures. CNN models (M) are explained in detail under subsection 5.3.1.1. The features from every CNN model are applied to a machine learning classifier, Cl. It results in the prediction of labels (y_i), for every test image.

5.3.1.1 Different CNN Architectures

CNN (Goodfellow *et al.* 2016) based DL models are VGG-16 (Simonyan & Zisserman 2014), VGG-19 (Simonyan & Zisserman 2014), Inception-V3 (Szegedy *et al.* 2016), Inception ResNet-V2 (Szegedy *et al.* 2016), Xception (Chollet 2017), MobileNet (Howard *et al.* 2017), DenseNet-121 (Huang *et al.* 2017), DenseNet-169 (Huang *et al.* 2017), and DenseNet-201 (Huang *et al.* 2017). These models are discussed in detail in the following subsections. In this research work, Global Average Pooling (GAP) and Fully Connected Layers (FCL) in all the above specified DL models are removed. Instead, machine learning classifiers are added to the DL models.

5.3.1.1.1 VGG-16 and VGG-19

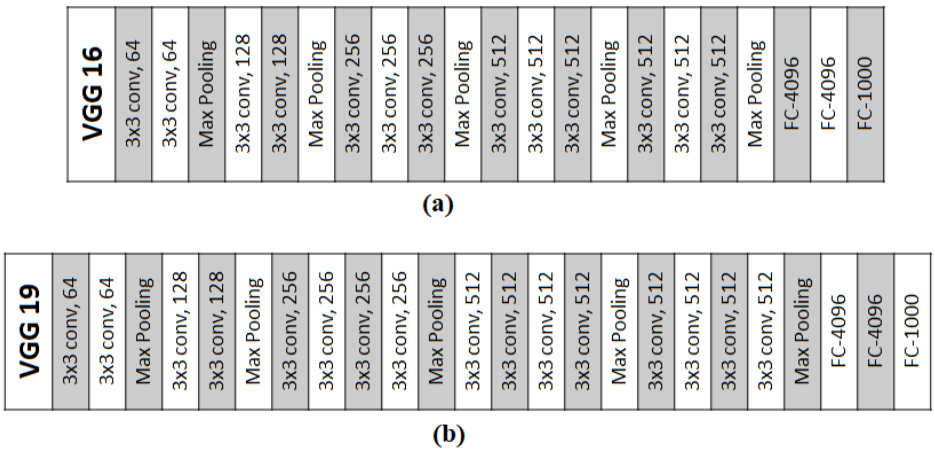


Figure 5.7 Layers of (a) VGG-16 CNN and (b) VGG-19 CNN

Simonyan & Zisserman (2014) introduced a very large Convolutional Neural Networks (VGG). VGG uses 3x3 convolution filters for deepening the network. VGG-16 and VGG-19 CNNs are a combination of convolution, max pooling, and three FCLs. Both the VGG CNN models utilize an increasing number of filters (64, 128, 256, 512). Its structure is shown in Figure 5.7.

5.3.1.1.2 ResNet-50

ResNet-50 (He *et al.* 2016) is a Residual Network with fifty layers. As the number of layers is linearly increased, the accuracy tends to get saturated. This problem is termed as Vanishing Gradient problem. ResNet utilizes the skip connections to solve the vanishing gradient problem. The layer arrangement of ResNet-50 CNN architecture is similar to that of VGG CNN architecture, except the usage of skip connections. It is shown in figure 5.8.

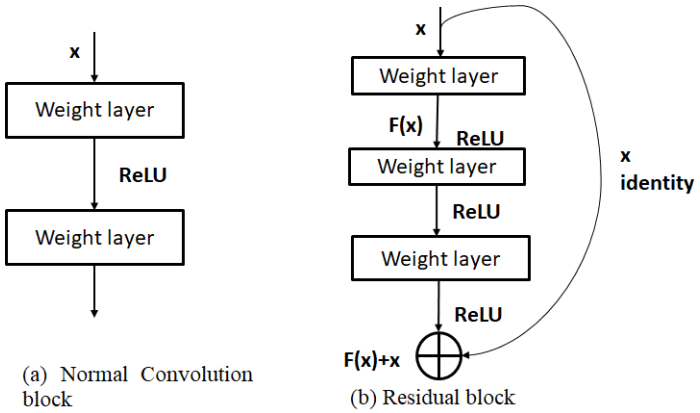


Figure 5.8 Difference between Normal Convolution Block and Residual Block

ResNet-50 utilizes 33 skip connections. A skip connection is introduced for every three layers (1x1, 3x3, 1x1) of convolution and max pooling. It is followed by an average pooling layer and 1000-d FCL with a Softmax activation function.

5.3.1.1.3 Inception-V3

For mobile and big data applications, high computational efficiency with less number of parameters is required. Factorized convolution and aggressive regularization are used to compute the optimized parameters. Inception-V3 (Szegedy et al. 2016) is an extension of GoogleNet CNN architecture (Szegedy et al. 2015).

Inception-V3 CNN architecture utilizes a 7 x 7 factorized convolutions. In every inception module, n x 1 filter is followed by 1 x n filter instead of n x n filters. This is termed as factorized convolution (k=7). The

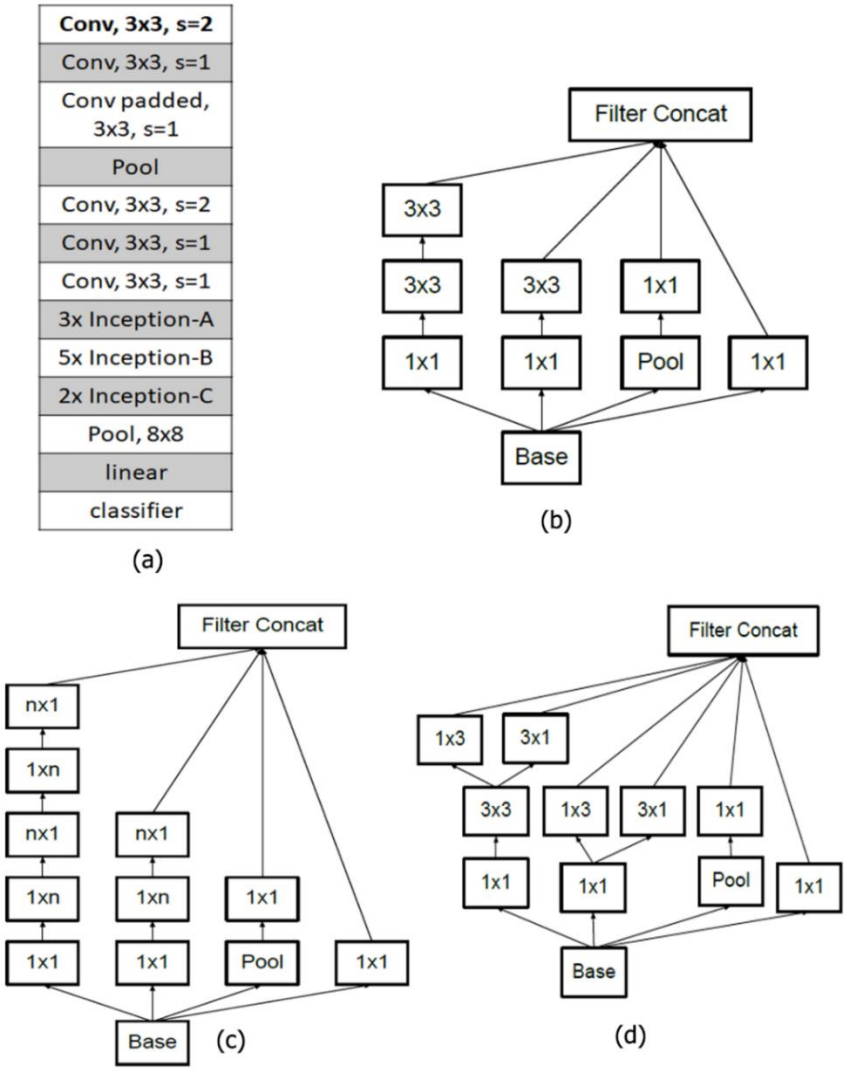


Figure 5.9 Inception-V3 architecture and modules (a) Architecture (b) Inception-A (c) Inception-B (d) Inception-C (Source: Szegedy *et al.* 2016)

architecture of Inception-V3 is shown in figure 5.9 (a). Figure 5.9 (b), (c), (d) shows the different inception modules with base block. The base block represents the feature maps obtained from previous layers.

5.3.1.1.4 Xception

Xception (Chollet 2017) means ‘Extreme Inception’, an extreme variant of Inception-V3. It comprises of 14 modules with 36 convolutional layers. Xception CNN architecture is shown in figure 5.10. The convolution and depthwise convolution are alternatively used in the Xception architecture. The difference between the normal convolution and depthwise convolution is discussed in subsection 5.3.1.1.6.

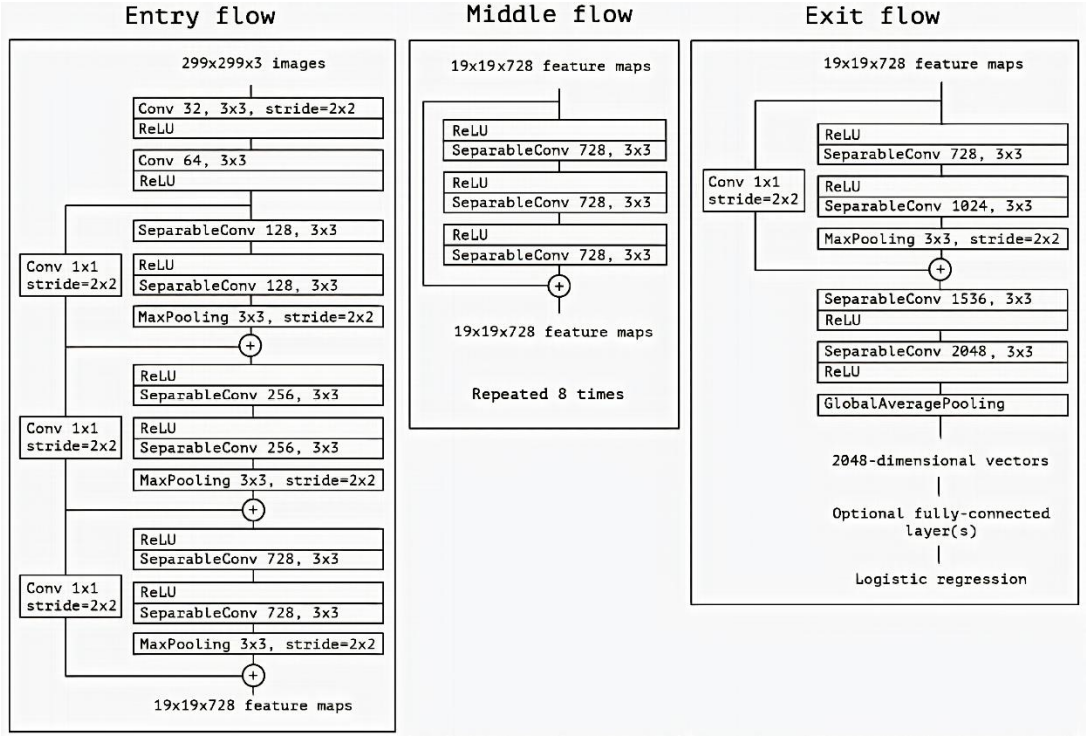


Figure 5.10 Xception Architecture (Source: Chollet 2017)

Xception utilizes the five residual connections, except for the first and last Xception modules. 1x1 Convolution is embedded within some of the Residual connections. The number of parameters in the Xception CNN model is similar to that of Inception-V3 CNN model. Comparatively, the Xception CNN model performs better than Inception CNN model.

5.3.1.1.5 Inception ResNet-V2

Inception-ResNet-V2 (Szegedy et al. 2016) CNN architecture combines the concepts of Inception and ResNet (He et al. 2016) architectures. Inception-ResNet-V2 contains stem (figure 5.11(b)) and inception blocks with residual connections, as shown in Figure 5.11(a).

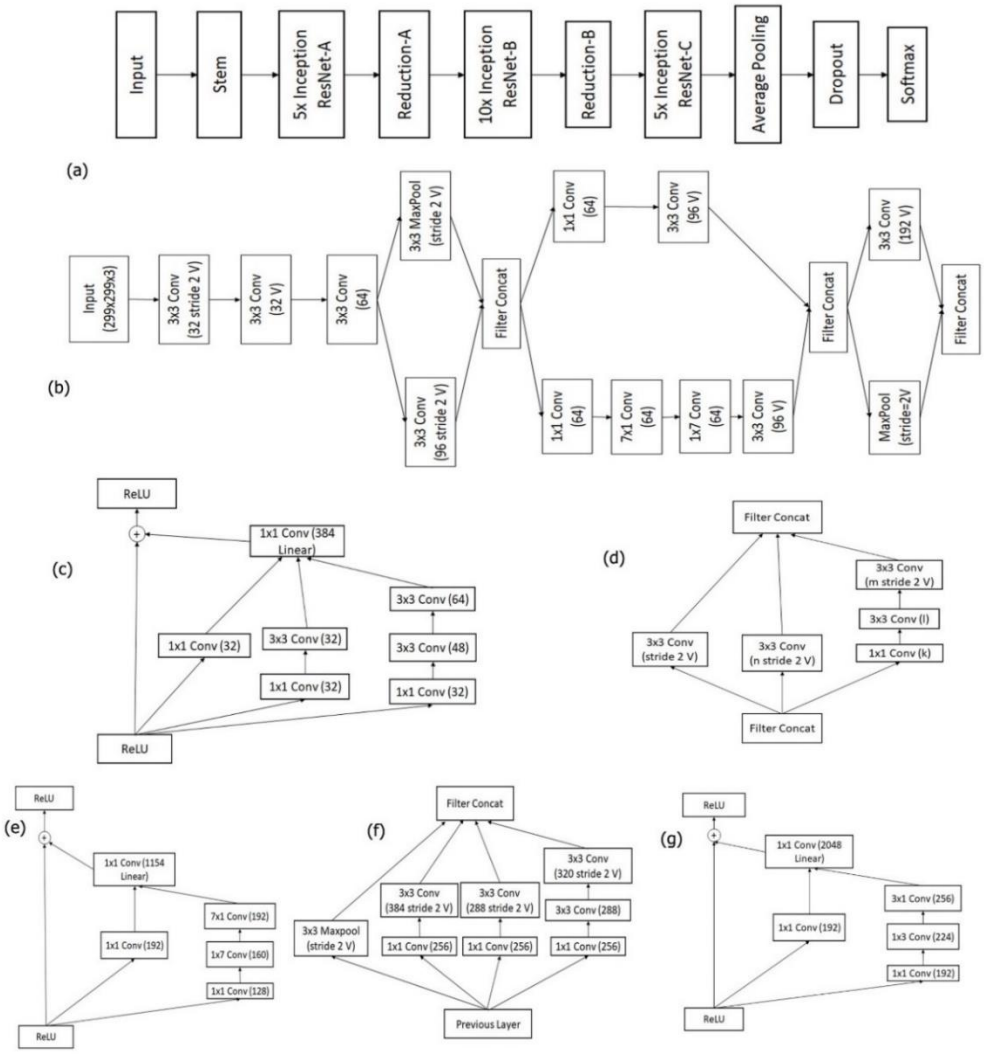


Figure 5.11 Inception ResNet-V2 (a) Architecture (b) Stem (c) Inception ResNet-A (d) Reduction-A (e) Inception ResNet-B (f) Reduction-B (g) Inception ResNet-C (Source: Szegedy *et al.* 2016)

Inception modules are present within the Residual connections, as shown in Figures 5.11 (c), (e), and (g). These are known as Inception ResNet

blocks. There are two reduction modules A and B, as depicted in figure 5.11 (d) and (f). Reduction modules are the same as inception block utilizing the factorized convolutions. A 1x1 convolution without activation function follows every inception block.

5.3.1.1.6 MobileNet

MobileNet (Howard *et al.* 2017) is a DL model with 28 layers. It is developed to embed a computer vision-based image classification application in mobile phones. It uses the Depthwise Separable Convolution (DWSC) similar to the Xception CNN model (Chollet 2017). In DWSC, a separable convolution is followed by pointwise (1x1) convolution. Batch normalization and ReLU (Rectified Linear Unit) activation function follow every DWSC or Normal Convolution (NC) process.

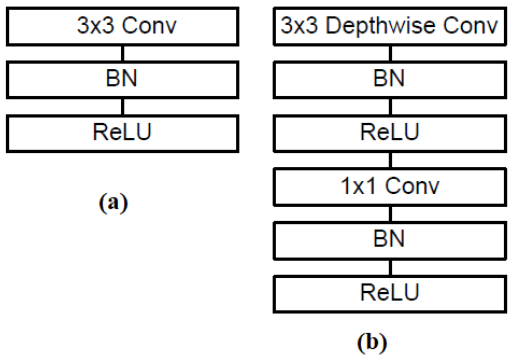


Figure 5.12 (a) Normal Convolution and (b) Depthwise Separable Convolution

Depthwise Separable Convolution (*DWSC*) consists of two layers, namely, depthwise convolution (*DWC*) and pointwise convolution (*PWC*). The difference between *DWSC* and *NC* is shown in figure 5.12. In DWC, the filters are individually applied to each color channel of the input image. Thus, the *DWC* produces a separate output feature map for each color channel. Equations (Kaiser *et al.* 2017) of Normal Convolution (*NC*), Depthwise Convolution

(*DWC*), Pointwise Convolution (*PWC*), and Depthwise Separable Convolution (*DWSC*) are given in Equations (5.3), (5.4), (5.5) and (5.6).

$$NC(K,f)_{(m,n)} = \sum_{i=1}^{i=I} \sum_{j=1}^{j=J} \sum_{u=1}^{u=U} K_{(i,j,u)} \cdot f_{(m+i,n+j,u)} \quad (5.3)$$

$$DWC(K_{dw},f)_{(m,n)} = \sum_{i=1}^{i=I} \sum_{j=1}^{j=J} K_{dw(i,j)} \odot f_{(m+i,n+j)} \quad (5.4)$$

$$PWC\left(K_{pw},f\right)_{(m,n)} = \sum_{u=1}^{U} K_u \cdot f_{(m,n,u)} \quad (5.5)$$

$$DWSC(K_{pw},K_{dw},f)_{(m,n)} = PWC_{(m,n)}(K_{pw},DWC_{(m,n)}(W_{dw},f)) \quad (5.6)$$

K, K_{pw}, K_{dw} are the filters of *NC*, *PWC*, and *DWC* containing weights, where *i*=1 to *I* rows (width of the image), *j*=1 to *J* columns (height of the image), and there are *U* number of filters (*u*=1 to *U*). The image is given by *f(m,n)* whose *(m, n)* represents the index position of the image. *PWC* is 1x1 convolution, used to merge the outputs of *DWC*.

5.3.1.1.7 DenseNet

DenseNet CNN architecture (Huang *et al.* 2017) consists of four dense blocks, as shown in figure 5.13(a). The transition layer is embedded in between every two dense blocks. The Global Average Pooling Layer and a Fully Connected Layer, follows the fourth dense block. The dense block comprises of 3x3 and 1x1 convolution sets, repeated multiple times in each block. Figure 5.13(b) shows a Dense block-1, whose dense connections are visualized only for the first 1x1 convolution. The transition layer consists of 1x1 convolution and 2x2 average pooling with a stride value of two. The transition layer is shown in Figure 5.13 (c).

DenseNet CNN architecture uses a hyperparameter called 'Growth Rate', represented as 'ρ'. The convolutions within the Dense block connects every layer to every other layer in the network. The dense connections provide a feature reuse capability with a reduced number of parameters. These dense

connections follow a feed-forward mechanism between the layers. It is represented in Equation (5.7). Also, the DenseNet CNN architecture solves the issue of vanishing gradient, similar to ResNet CNN architecture. Hence, it is considered as a variant of ResNet CNN architecture.

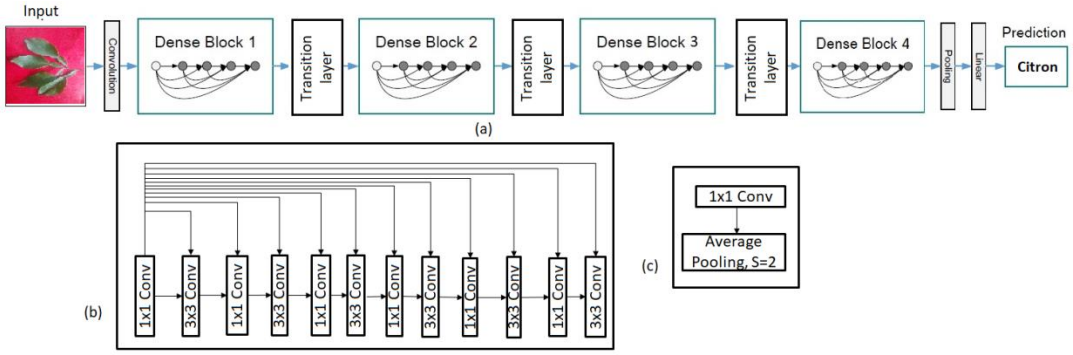


Figure 5.13 DenseNet Architecture (a) Architecture (b) Dense block (c) Transition layer

$$a_L = D_L([a_0, a_1, \dots, a_{L-1}]) \qquad (5.7)$$

where a_L is the current layer and a concatenation of feature maps of all previous layers $a_0, a_1, .., a_{L-1}$. $D_L(.)$ is a composite function consisting of three consecutive operations, namely, Batch Normalization (BN), ReLU, and 3x3 convolution.

5.3.1.2 Machine Learning Classifiers

Global Average Pooling (GAP) and Fully Connected Layers (FCLs) in the DL architectures are replaced with machine learning classifiers. The machine learning classifiers used are multinomial Logistic Regression (LR), Linear Discriminant Analysis (LDA), Naïve Bayes (NB), K-Nearest Neighbor (K-NN), Classification and Regression Tree (CART), Random Forest Classifier (RFC), Bagging Classifier (BC), Support Vector Machine (SVM) and Multi-Layer Perceptron (MLP) (Marsland 2014; Mitchell 1997). Except for MLP, all the other classifiers are discussed in chapter 3.

5.3.1.2.1 Multi-Layer Perceptron (MLP)

MLP (Marsland 2014; Mitchell 1997) is an Artificial Neural Network model. It follows the Backpropagation algorithm. In this research work, one hidden layer with 512 neurons is considered. For MLP, the results are obtained by using Adam optimizer with a learning rate of 0.001. The number of epochs is set to 200. A parameter 'Tol' is set to 0.0001. 'Tol' parameter checks for the convergence, i.e., the iterations stop if ten consecutive iterations do not show at least an improvement of 0.0001 (Pedregosa *et al.* 2005).

5.3.2 Approach-2

The DL models explained in subsection 5.3.1.1 are utilized. Except for VGG-16 and VGG-19 CNN architectures, all the other M models use one GAP and one FCL. Both VGG architectures use three FCLs. GAP and FCLs in DL architectures are removed. They are replaced with a FCL having a softmax activation function. The number of neurons in the FCL is the same as that of the number of classes in the dataset. The steps involved for training a fine-tuned CNN architecture is represented as an Algorithm 5.2.

Algorithm 5.2: Training of Fine-tuned CNN Architecture

Input: Images, I_N; ImageNet weights, *IW*; ImageNet biases, *IB*; Model, *M*; η=0.0001; Epochs, *ep*=50; Batch Size, *BS*=32

Output: Predicted labels, y_i

Step 1: Read the *N* number of images (I_N)

Step 2: Resize the I_N images to 100x100x3

Step 3: Pass I_N through the CNN model, *M*

Step 4: Train the CNN model, *M* using Adam optimizer

for i=1 to M:

for j=1 to ep:

Update weights, IW and biases, IB

End

End

For the convolution base (layers in the model, *M*), the pre-trained ImageNet weights are used as initial weights. On adding the FCL, the M^{th} CNN model is trained from the convolution base. The training parameters are 50 epochs, learning rate of 0.0001 with Adam optimizer, and a batch size of 32. Since the learning rate of 0.001 resulted in poor loss and accuracy values, it is reduced to 0.0001. A learning rate of 0.00001 makes the model *M* to learn the data at a slow rate. Also, the obtained accuracies are lesser in comparison to the accuracies obtained for a learning rate of 0.0001, irrespective of *M*.

5.4 RESULTS AND DISCUSSION

The performance analysis of different deep learning architectures for plant species recognition is carried out. Flavia (Wu *et al.* 2007), Folio, Swedish Leaf (Soderkvist 2002), and custom-developed Leaf-12 datasets are used in the studies. The results obtained are tabulated and discussed in the subsections (5.4.1-5.4.4). The train-test data split ratio is 70:30. The experimentation is carried out in Python 3.5 along with Keras (Tensorflow as backend) package. Also, few supporting packages such as Scikit-learn, H5py, Numpy, OS, Matplotlib, and Seaborn packages are used. The hardware setup includes Windows 10 64-bit OS with Intel Core i7-4790 CPU with 20GB RAM. Additionally, NVIDIA Titan X GPU with 3584 CUDA cores is interfaced with the CPU through the PCI interface.

5.4.1 Flavia dataset

Performance (Top-1 accuracy) of different DL architectures for Flavia dataset are listed in Table 5.3. Results obtained by the two approaches

(Approach 1: CNN as feature extractor + Machine Learning classifier, Approach 2: Fine-tuned CNN architectures) of experimentation are shown in the table. Irrespective of the Deep Learning architectures, the results obtained by approach 2 (Fine-tuned CNN architecture) is better compared to approach 1. The DenseNet CNN architecture outperformed other CNN architectures. It is also observed that the fine-tuned DenseNet-169 CNN obtained the highest Top-1 accuracy (98.58%) for plant species recognition using Flavia dataset.

Table 5.3 Accuracies obtained by Deep Learning Architectures (%)

CNN Model	CNN as feature extractor + Machine Learning classifier									Fine-tuned CNN
	Machine Learning classifier									
	LDA	LR	NB	KNN	CART	RFC	BC	MLP	SVM	
VGG16	71.67	95.00	87.92	85.42	66.25	92.50	84.58	94.38	95.62	**96.88**
VGG19	59.17	95.00	85.62	92.92	69.17	92.50	85.62	93.12	95.21	**97.29**
Inception-V3	66.04	91.67	53.33	82.92	46.04	85.21	71.25	90.21	89.79	**97.50**
ResNet50	88.96	97.29	93.12	94.79	71.67	93.75	85.83	95.83	97.50	**98.12**
Inception ResNet-V2	49.17	92.71	58.54	78.12	58.33	84.38	75.21	92.92	89.79	**98.17**
Xception	81.25	92.71	64.38	82.92	58.75	86.88	76.88	87.29	91.25	**98.17**
MobileNet	60.00	96.67	93.12	95.00	68.12	95.21	86.25	96.67	96.25	**97.29**
DenseNet-121	80.42	97.62	92.29	94.58	70.62	96.88	87.92	96.46	96.67	**98.12**
DenseNet-169	83.12	97.92	92.50	94.58	73.54	95.62	88.12	98.12	97.50	**98.58**
DenseNet-201	85.42	97.92	93.12	94.17	71.46	96.04	91.04	98.12	97.92	**98.38**

Other than DenseNet CNN models, the ResNet-50 (98.12%), Inception ResNet-V2 (98.17%) and Xception (98.17%) CNN architectures resulted in an accuracy of 98% or greater.

In approach 1 (CNN as feature extractor + ML classifiers), the classifiers such as Logistic Regression, Multi-Layer Perceptron, Support Vector

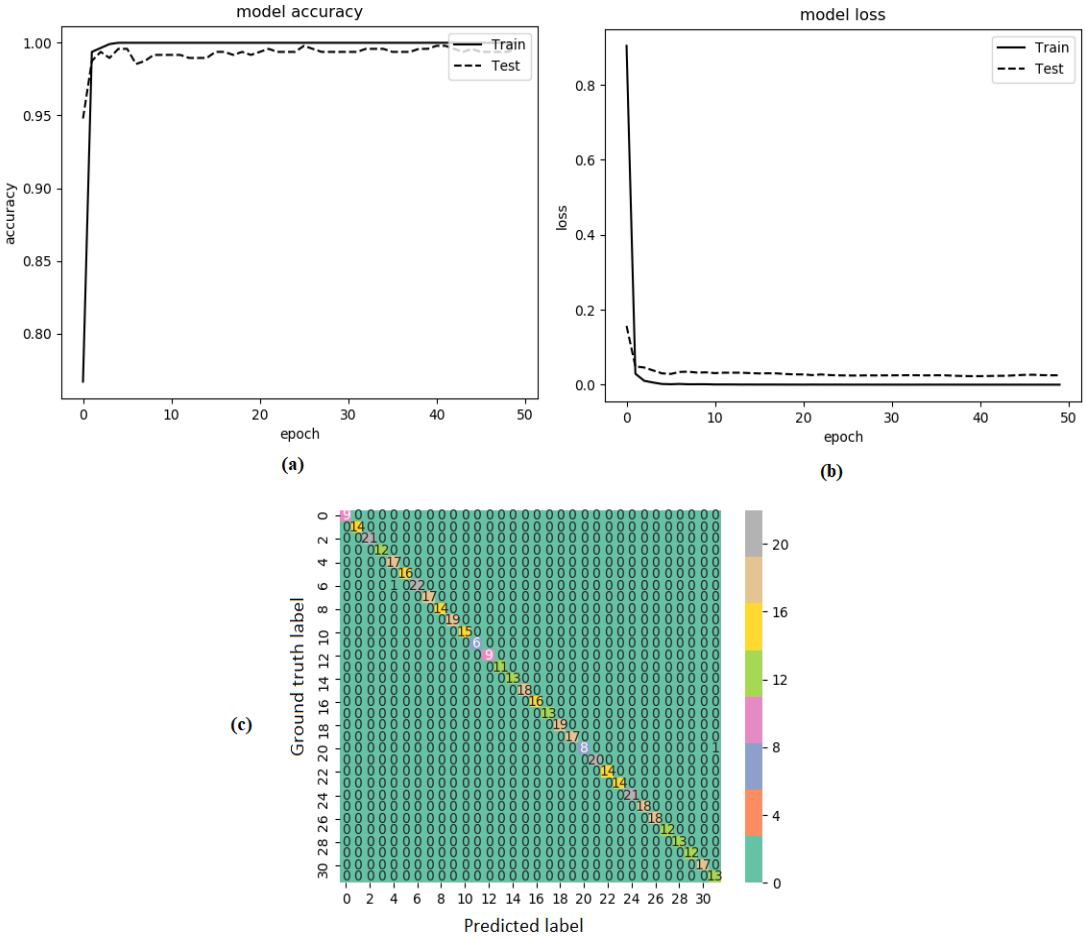


Figure 5.14 Performance of the Fine-tuned DenseNet-169 CNN Architecture (a) Accuracy plot (b) Loss graph and (c) Confusion Matrix

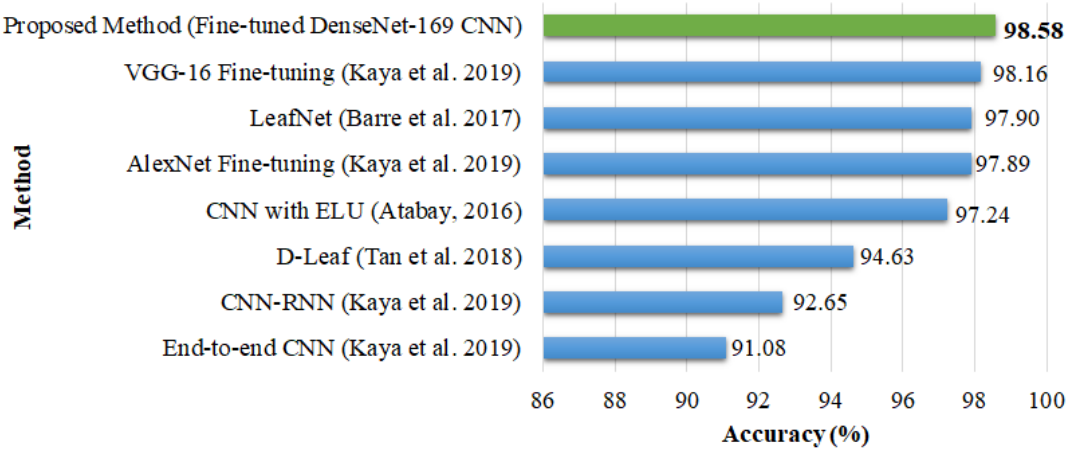


Figure 5.15 Comparison of results by existing methods with the fine-tuned DenseNet-169 CNN architecture

Machine perform comparatively better to other classifiers. The DenseNet-201 CNN architecture resulted in higher accuracies (LR- 97.92%, MLP- 98.12%, SVM- 97.92%).

The performance plot for Fine-tuned DenseNet-169 CNN architecture is shown in figure 5.14 (a)-(c). It is observed that the CNN model does not overfit the data. Fine-tuned DenseNet-169 gets stabilized after 10 epochs. The loss curve for the DenseNet-169 CNN architecture is shown in figure 5.14(b). It is observed that the loss values are greatly reduced within 50 epochs, resulting in improved accuracy for the PSR system. The confusion matrix is shown in figure 5.14 (c). It indicates that the number of mispredictions is greatly reduced when compared to Neural Networks and Conventional methods.

The comparison results of the existing and the proposed methods are shown in figure 5.15. It is observed that the proposed method (fine-tuned DenseNet-169 CNN architecture) resulted in higher accuracy (Top-1: 98.58%) compared to other existing literatures.

5.4.2 Folio dataset

Table 5.4 shows the accuracies obtained by using two approaches (Approach 1: CNN as feature extractor + ML classifiers, Approach 2: Fine-tuned CNN architectures). It is observed that the DenseNet CNN architectures resulted in higher accuracies compared to other CNN architectures.

In approach 1 (CNN as feature extractor + ML classifier), two combinations produced higher accuracy of 95.95%. The combinations are DenseNet-121 + Logistic Regression (LR) classifier and DenseNet-169 + Multilayer perceptron (MLP) method. It is also observed that the Logistic Regression (LR), Multi-layer perceptron (MLP), and Support Vector Machine (SVM) classifiers performed better compared to other classifiers.

In approach 2 (Fine-tuned CNN architectures), the two DenseNet CNN architectures resulted in similar accuracy (95.95%) as that of approach 1. Each DenseNet CNN architecture comprises 169 and 201 layers.

Table 5.4 Accuracies obtained by Deep Learning Architectures (%)

CNN Model	CNN as feature extractor + Machine Learning classifier									Fine-tuned CNN
	Machine Learning classifier									
	LDA	LR	NB	KNN	CART	RFC	BC	MLP	SVM	
VGG16	83.24	**95.38**	84.39	90.17	54.34	89.02	75.72	90.17	91.33	93.64
VGG19	77.46	**95.38**	82.66	90.75	65.9	88.44	73.99	93.64	93.06	90.75
Inception-V3	82.08	90.75	53.18	81.50	34.68	73.41	49.71	86.71	85.55	**91.91**
ResNet50	84.97	**94.80**	90.75	90.75	60.12	90.17	80.35	90.17	94.22	**94.80**
Inception ResNet-V2	82.66	85.55	58.38	74.57	39.88	78.61	59.54	84.97	80.35	**95.53**
Xception	83.24	88.44	70.52	77.46	46.82	75.14	62.43	83.82	84.97	**95.38**
MobileNet	86.71	**92.49**	88.44	89.60	50.29	87.86	73.99	87.86	91.33	86.13
DenseNet-121	91.91	**95.95**	86.13	91.33	61.27	90.75	77.46	91.33	93.06	92.49
DenseNet-169	94.22	95.53	88.44	90.75	66.47	93.06	82.66	**95.95**	94.80	**95.95**
DenseNet-201	93.64	94.80	86.71	89.60	67.05	91.91	84.39	90.75	93.64	**95.95**

When compared to Flavia dataset, lower performance of the DL architectures on Folio dataset is observed. This is due to the less number of images in the dataset for training and testing. Also, the number of images per class is lesser compared to the Flavia dataset.

Figure 5.16 shows the performance curves for DenseNet CNN architecture. The plots (a) and (b) indicate the accuracy and loss obtained by using fine-tuned DenseNet-169 CNN architecture. Plots (c) and (d) show the accuracy and loss curves for DenseNet-201 architecture. From the plots, it is observed that accuracy and loss curves get stabilized from the 5th epoch.

Figure 5.17 (a)-(d), shows the confusion matrix obtained for both the approaches (Approach 1: Plot (c) and (d), Approach 2: Plot (a) and (b)). Plot (a)

and (b) represents the confusion matrix obtained for fine-tuned DenseNet-169 and DenseNet-201 CNN architectures. Plot (c) and (d) represents the confusion

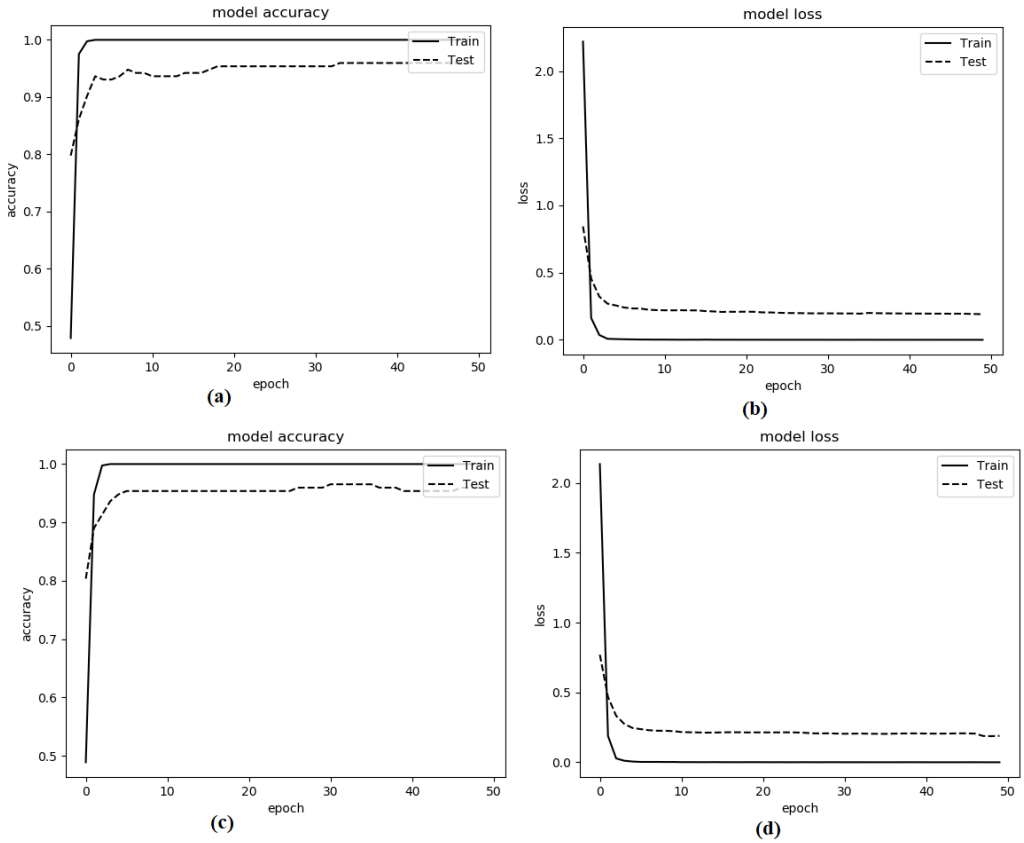


Figure 5.16 Performance of DL Architecture (a) Accuracy plot for Fine-tuned DenseNet-169 CNN (b) Loss curve for Fine-tuned DenseNet-169 CNN (c) Accuracy plot for Fine-tuned DenseNet-201 CNN (d) Loss curve for Fine-tuned DenseNet-201

matrix obtained for DenseNet-121 + LR and DenseNet-169 + MLP CNN architectures. From the plots, it is observed that the number of mispredictions in fine-tuned DenseNet CNN architectures is less compared to DenseNet + ML classifiers. Hence, fine-tuned DenseNet CNN architecture produced higher accuracies.

The comparison results of the existing and the proposed methods for the Folio dataset are shown as a plot in figure 5.18. It is observed that the proposed method (DenseNet-121 + LR, DenseNet-169 + MLP, fine-tuned

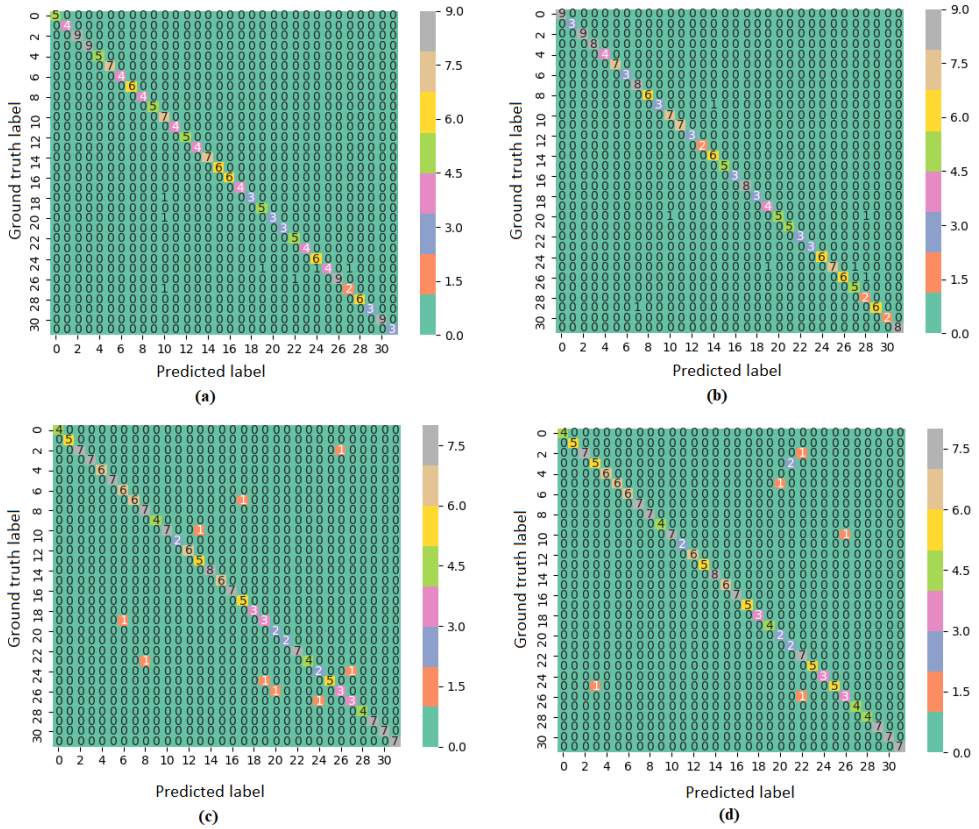


Figure 5.17 Confusion Matrices obtained by both the approaches (a) Fine-tuned DenseNet-169 CNN (b) Fine-tuned DenseNet-201 CNN (c) DenseNet-121+LR classifier (d) DenseNet-169+MLP classifier

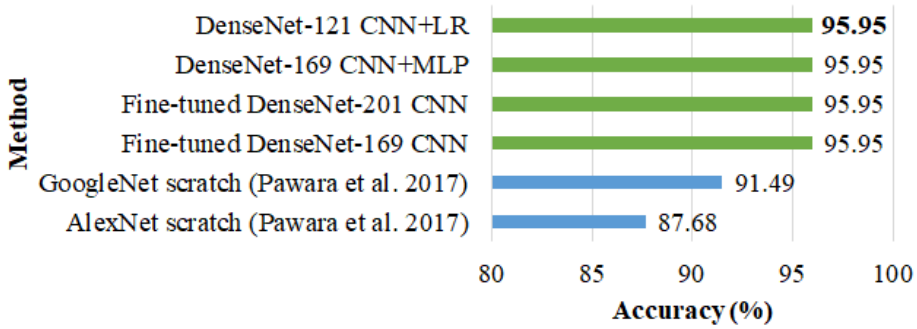


Figure 5.18 Comparison of results by Existing methods with DenseNet CNN Architectures

DenseNet-169 CNN, fine-tuned DenseNet-201 CNN) showed higher accuracy (Top-1: 95.95%) compared to other existing literatures.

5.4.3 Swedish leaf dataset

Top-1 accuracy for various DL architectures with the Swedish leaf dataset is listed in Table 5.5. Swedish leaf dataset attained an accuracy of 99.41% for fine-tuned DenseNet-121 CNN (Approach 2), fine-tuned DenseNet-169 (Approach 2), and MobileNet CNN+LR (Approach 1) architectures.

Table 5.5 Accuracies obtained by Deep Learning Architectures (%)

CNN Model	CNN as feature extractor + Machine Learning classifier									Fine-tuned CNN
	Machine Learning classifier									
	LDA	LR	NB	KNN	CART	RFC	BC	MLP	SVM	
VGG16	89.05	98.52	93.79	96.75	81.95	96.45	93.79	97.04	98.22	**99.00**
VGG19	86.09	99.41	91.72	95.27	79.59	96.75	90.53	97.33	99.11	**99.20**
Inception-V3	86.39	94.67	68.93	89.05	68.05	88.17	80.18	93.78	93.20	**99.11**
ResNet50	96.75	99.11	94.38	96.75	81.07	97.63	95.56	97.34	99.1	**99.30**
Inception ResNet-V2	78.99	96.15	68.34	83.43	71.01	89.05	86.09	95.26	94.67	**98.22**
Xception	90.53	96.45	79.88	89.35	72.19	92.01	84.91	95.85	96.15	**99.31**
MobileNet	94.67	**99.41**	91.72	95.86	80.47	95.86	93.49	98.22	99.30	97.04
DenseNet-121	93.20	98.93	85.21	94.97	81.66	98.52	95.27	98.52	98.22	**99.41**
DenseNet-169	86.98	98.52	96.75	95.27	84.62	98.22	96.75	97.92	98.82	**99.41**
DenseNet-201	85.80	98.82	96.75	97.63	84.32	98.52	93.79	98.22	98.52	**99.31**

Figure 5.19 shows the performance curves for DenseNet CNN architecture. The plots (a) and (b) indicate the accuracy and loss plots obtained by using DenseNet-121 CNN architecture. Plots (c) and (d) show the accuracy and loss curves for DenseNet-169 architecture. From the plots, it is observed that the loss curve for DenseNet-169 CNN architecture stabilizes after 20 epochs compared to that of DenseNet-121 CNN architecture. With the minimum loss, higher accuracy is achieved.

Figure 5.20 (a)-(c), shows the confusion matrix obtained for both the approaches (Approach 1: Plot (c), Approach 2: Plot (a) and (b)). Plot (a) and (b)

represents the confusion matrix obtained for fine-tuned DenseNet-121 and DenseNet-169 CNN architectures. Plot (c) represents the confusion matrix obtained for MobileNet CNN + LR architecture. From the plots, it is observed that the number of mispredictions in Fine-tuned DenseNet CNN architectures is

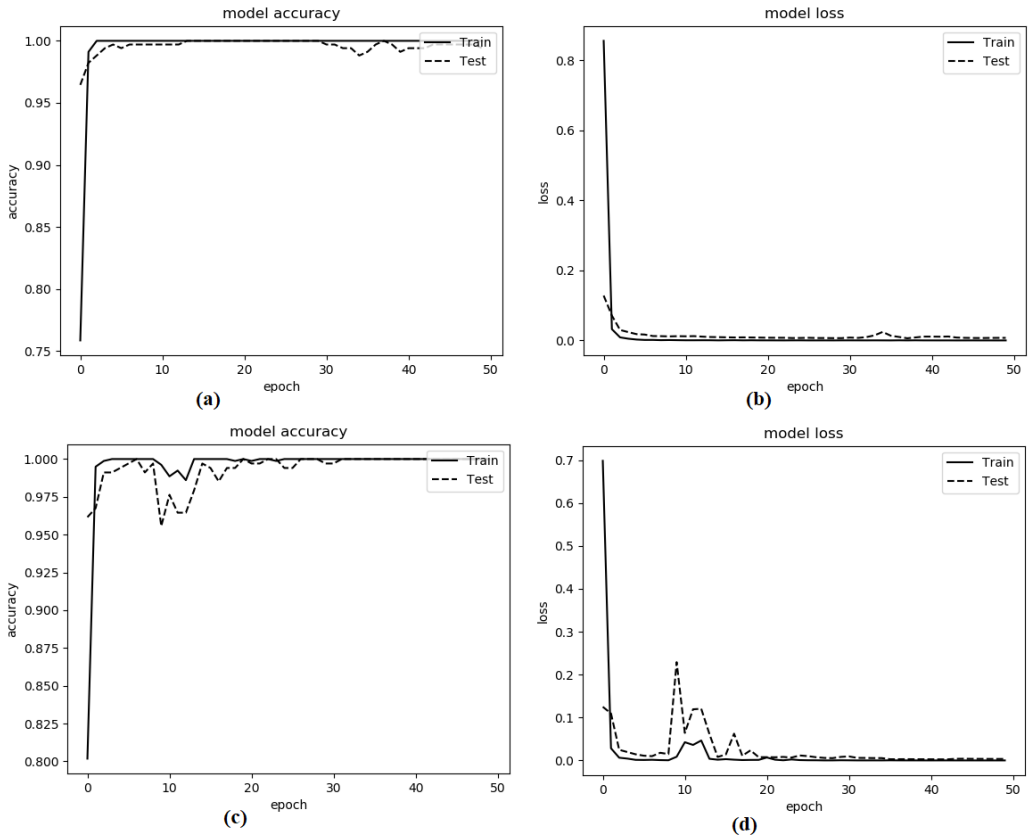


Figure 5.19 Performance Curves for DL architectures (a) Accuracy plot for fine-tuned DenseNet-121 CNN and (b) Loss Curve for fine-tuned DenseNet-121 CNN (c) Accuracy plot for fine-tuned DenseNet-169 CNN (d) Loss Curve for fine-tuned DenseNet-169 CNN

less (almost nil mispredictions) compared to MobileNet CNN + LR classifier (1 misprediction). Thus, comparable accuracies are achieved for both approaches.

The comparison results between the existing and the proposed method for the Swedish Leaf dataset are shown as a plot in figure 5.21. It is observed that the proposed method (MobileNet CNN+LR, fine-tuned DenseNet-121 CNN, fine-tuned DenseNet-169 CNN) resulted in higher

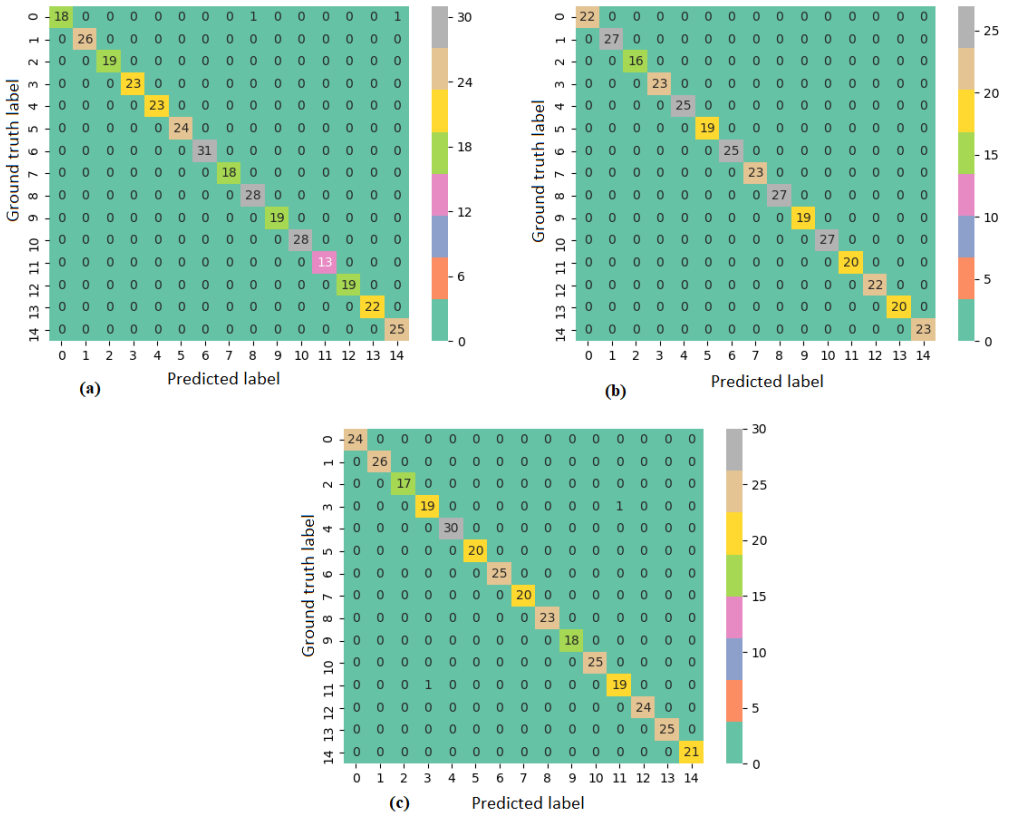


Figure 5.20 Confusion Matrices obtained for DL architectures (a) Fine-tuned DenseNet-121 CNN (b) Fine-tuned DenseNet-169 CNN (c) MobileNet CNN+LR architecture

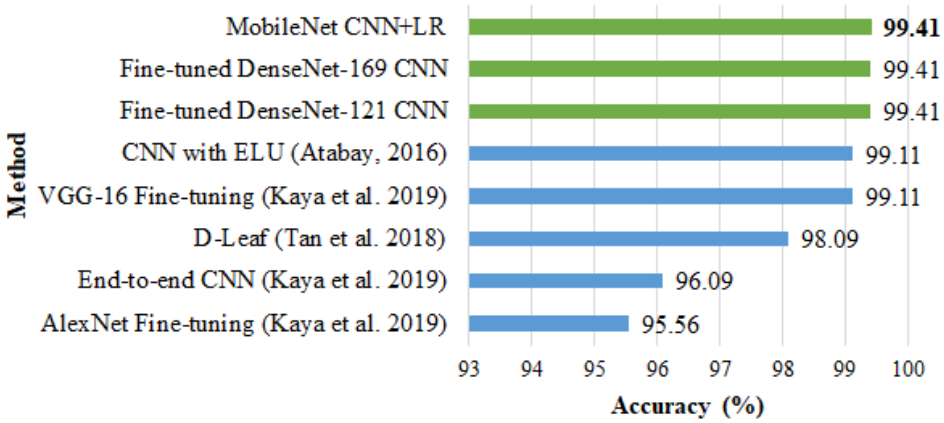


Figure 5.21 Comparison of results by existing methods with the Proposed CNN architectures

accuracy (Top-1: 99.41%) compared to other existing literatures. The higher plant species recognition rate is due to the quality of the dataset (More pre-processed images).

5.4.4 Leaf-12 dataset

Table 5.6 Accuracies obtained by Deep Learning Architectures (%)

CNN Model	CNN as feature extractor + Machine Learning classifier									Fine-tuned CNN
	Machine Learning classifier									
	LDA	LR	NB	KNN	CART	RFC	BC	MLP	SVM	
VGG-16	79.60	97.22	64.84	91.23	63.80	92.88	79.77	94.27	96.70	**98.44**
VGG-19	79.77	96.18	63.63	90.80	61.20	93.14	76.65	94.36	96.27	**98.09**
Inception-V3	67.71	90.28	47.31	83.07	42.62	81.77	61.37	90.80	91.06	**97.92**
ResNet50	95.31	97.92	48.78	93.06	62.67	92.88	78.30	96.09	98.00	**98.78**
Inception ResNet-V2	89.32	93.75	60.42	85.85	51.56	85.07	72.92	93.40	93.23	**98.87**
Xception	92.45	94.36	66.93	87.76	56.34	85.68	72.83	94.10	94.53	**98.26**
MobileNet	96.09	97.14	59.11	94.01	61.63	93.40	81.77	97.14	97.48	**98.52**
DenseNet-121	97.22	97.92	89.24	95.14	68.23	95.40	84.46	97.14	97.83	**98.26**
DenseNet-169	97.66	97.66	89.32	95.92	64.06	95.49	84.29	97.74	97.57	**99.00**
DenseNet-201	**98.61**	97.92	92.01	96.27	66.41	96.88	82.20	97.48	98.00	98.26

Performance metric (Top-1 accuracy) for the PSR system using different DL architectures is listed in Table 5.6. From the table data, it is observed that the fine-tuned DL architectures (Approach 2) performed better (>98%) compared to DL architecture + ML classifiers (Approach 1), irrespective of the architectures considered. In approach 1, the DenseNet-201 CNN + LDA classifier resulted in higher accuracy (98.61%) compared to other

methods. In approach 2, the fine-tuned DenseNet-169 CNN architecture resulted in higher accuracy (99.00%) compared to other methods.

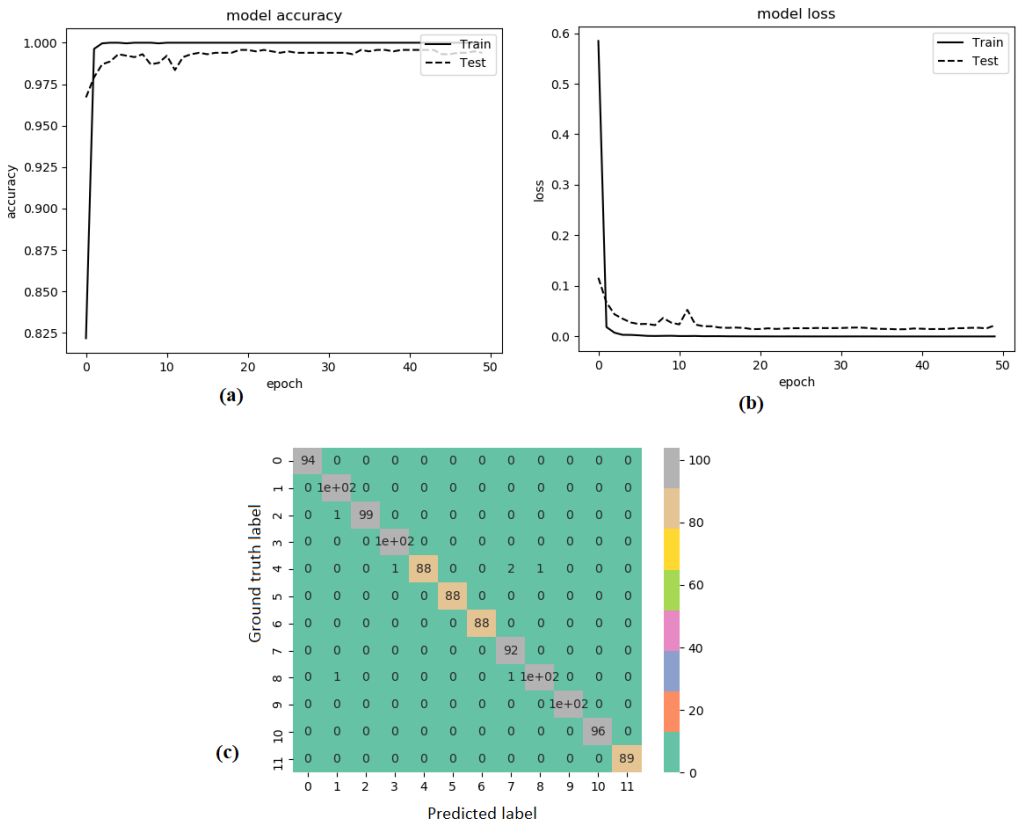


Figure 5.22 Performance curves for fine-tuned DenseNet-169 CNN architecture (a) Accuracy Plot (b) Loss Curve (c) Confusion Matrix

Figure 5.22 (a) and (b), shows the accuracy plot and loss curve obtained for fine-tuned DenseNet-169 CNN architecture. From the plots, it is observed that the loss is minimized and stabilized after 15 epochs. Plot (c) represents the confusion matrix obtained for fine-tuned DenseNet-169 CNN architecture. From the plots, it is observed that there is less number of mispredictions in fine-tuned DenseNet-169 CNN architecture, resulting in higher accuracies for the PSR system.

5.4.5 Results obtained by Adding Global Average Pooling (GAP) layer in DL Architectures

All the previously reported results are obtained without a Global Average Pooling (GAP) layer in the DL architectures. It resulted in the usage of non-dimensionally reduced feature maps for final classification.

The inclusion of a GAP layer in DL architectures before the final classification step reduces the predictive accuracy for both approaches (Approach 1: CNN as feature extractor + ML classifier, Approach 2: fine-tuned CNN architecture). The results obtained by adding a GAP layer in DL architecture is listed in Table 5.7. All four datasets are used for the analysis.

Table 5.7 Comparison of Results obtained by adding a GAP layer in DL Architecture with DL Architecture without GAP layer

Dataset	Model	Accuracy (%)	
		Without GAP	With GAP
Flavia	DenseNet-169 + LR	**97.92**	97.08
	Fine-tuned DenseNet-169 CNN	**98.58**	98.00
Folio	DenseNet-121 + LR	**95.95**	95.38
	Fine-tuned DenseNet-169 CNN	**95.95**	95.38
Swedish leaf	DenseNet-169 + LR	**98.52**	98.18
	Fine-tuned DenseNet-169 CNN	**99.23**	99.12
Leaf-12	DenseNet-169 + LR	**97.66**	97.40
	Fine-tuned DenseNet-169 CNN	**99.00**	97.31

It is observed from the table data that the DL architecture (without GAP layer) performs better than the DL architecture with GAP layer. The findings for all four datasets are similar (Flavia, Folio, Swedish leaf, and Leaf-12).

5.5 SUMMARY

Plant species recognition is one of the challenging tasks in computer vision due to the large diversity of species (intra-class and inter-class variations). In this chapter, the Convolutional Neural Networks based DL models (VGG-16, VGG-19, Inception-V3, ResNet-50, Inception-ResNet-V2, Xception, MobileNet, DenseNet-121, DenseNet-169, DenseNet-201) are utilized in the context of plant species recognition.

Table 5.8 Summary of Results (best) obtained by DL Architectures for the Four Datasets

Dataset	CNN Model	Accuracy (%)		Performance Metrics			Time (s)	
		Top-1	Top-5	Pr	Re	F1	Train and Preprocess	Test
Flavia	Fine-tuned DenseNet-169 CNN	98.58	99.79	0.99	0.98	0.98	422.82	3.53
Folio	**DenseNet-121 + LR**	**95.95**	**99.42**	**0.96**	**0.96**	**0.96**	**52.76**	**0.01**
	DenseNet-169 + MLP	95.95	99.42	0.97	0.96	0.96	209.81	0.10
	Fine-tuned DenseNet-169 CNN	95.95	98.27	0.97	0.96	0.96	188.29	3.12
	Fine-tuned DenseNet-201 CNN	95.95	97.69	0.97	0.96	0.96	227.33	3.96
Swedish leaf	**MobileNet + LR**	**99.41**	**100**	**0.99**	**0.99**	**0.99**	**66.5**	**0.01**
	Fine-tuned DenseNet-121 CNN	99.41	100	0.99	0.99	0.99	296.29	2.43
	Fine-tuned DenseNet-169 CNN	99.41	100	0.99	0.99	0.99	313.13	3.41
Leaf12	Fine-tuned DenseNet-169 CNN	99	100	0.99	0.99	0.99	996.02	4.45

Two approaches are followed to evaluate the DL architectures. In approach 1, the CNN architectures are used as feature extractors. Later, the

classification is performed by using the ML classifiers. In approach 2, the fine-tuned CNN architectures are utilized for plant species recognition.

Table 5.8 summarizes the best results obtained for the four datasets (Flavia, Folio, Swedish leaf, Leaf-12). For Flavia dataset, the fine-tuned DenseNet-169 CNN architecture performed better compared to other methods. The performance metrics obtained are: *Top-1* accuracy- 98.58%, Top-5 accuracy- 99.79%, Precision (*Pr*) - 0.99, Recall (*Re*) - 0.98 and F1-score (*F1*) - 0.98.

For Folio dataset, four models namely, fine-tuned DenseNet-169 CNN, fine-tuned DenseNet-201 CNN, DenseNet-121 + LR and DenseNet-169 + MLP, resulted in similar accuracy of 95.95%. For DenseNet-121 + LR, the time taken to obtain the results is lesser compared to fine-tuned DenseNet-169 CNN architecture. It is concluded that the DenseNet-121 + LR outperformed the other three models in terms of higher accuracy with lower computational time. The other performance metrics (Precision, Recall, and F1-score) are represented in Table 5.8. The accuracies obtained for the Folio dataset is lower compared to the other three datasets. This might be due to the less number of image samples in the dataset (18 images/class).

For the Swedish leaf dataset, the fine-tuned DenseNet-121 CNN, fine-tuned DenseNet-169 CNN, and MobileNet+LR resulted in similar *Top-1* (99.41%) and *Top-5* (100%) accuracies. The performance metrics (Precision, Recall, and F1-score) are the same for the three models. The approach-1 (MobileNet+LR) performs better in terms of higher accuracy with lower computational time.

For the Leaf-12 dataset, the fine-tuned DenseNet-169 CNN architecture resulted in higher accuracy (99%) compared to all the other methods. The performance metrics (Precision, Recall, and F1-score) of fine-tuned DenseNet-169 CNN architecture are listed in Table 5.8.

From the results of DL architectures, it is observed that the DenseNet CNN architecture, either in approach 1 (CNN as feature extractor + ML classifier) or approach 2 (fine-tuned CNN architecture), performed better compared to other CNN architectures. This might be due to the frequency reuse property of the DenseNet CNN architecture. Also, it is observed that the computation time of approach 2 is longer compared to approach 1.

The performance of single deep learning architectures is better compared to Neural Network and Traditional methods. The main disadvantage of single Deep Learning architecture is in the increased number of mispredictions when it is exposed to real-time data or images. The above-specified criterion is minimized by the usage of Dual Deep Learning architectures. The methodology and results obtained for Dual Deep Learning architectures are discussed in detail in the next chapter.

CHAPTER 6

DUAL DEEP LEARNING ARCHITECTURES FOR PLANT SPECIES RECOGNITION

6.1 INTRODUCTION

In the previous chapter, the performance of individual Deep Learning CNN models as feature extractor and classifier are reported. The CNN architectures considered in chapter 5 are VGG-16, VGG-19, Inception-V3, Inception-ResNet-V2, Xception, ResNet50, DenseNet-121, DenseNet-169, DenseNet-201 and MobileNet.

The accuracies obtained for individual Deep Learning architectures for plant species recognition are summarized in Table 6.1. The performance analysis is carried out by using four datasets (Flavia, Folio, Swedish Leaf, and custom-developed Leaf-12).

Table 6.1 Accuracies of Single Deep Learning Architectures (%)

Dataset	CNN Model	Accuracy (%)	
		Top-1	Top-5
Flavia	Fine-tuned DenseNet-169 CNN	98.58	99.79
Folio	**DenseNet-121 + LR**	**95.95**	**99.42**
	DenseNet-169 + MLP	95.95	99.42
	Fine-tuned DenseNet-169 CNN	95.95	98.27
	Fine-tuned DenseNet-201 CNN	95.95	97.69
Swedish leaf	**MobileNet + LR**	**99.41**	**100**
	Fine-tuned DenseNet-121 CNN	99.41	100
	Fine-tuned DenseNet-169 CNN	99.41	100
Leaf12	Fine-tuned DenseNet-169 CNN	99	100

From the table, it is observed that the accuracies obtained by using single Deep Learning architecture are above 98% (Flavia, Swedish Leaf, and Leaf-12 datasets), except for the Folio dataset (95.95%). When the single deep

learning architectures are exposed to real-time images of the plant leaf, it resulted in more number of mispredictions. Also, the computation time of fine-tuned single deep learning architecture is high compared to CNN as feature extractor + Machine learning classifier method.

To minimize the number of mispredictions for real-time plant images and to lower the computation cost, Dual Deep Learning Architectures are proposed for plant species recognition. Based on the characteristics of single deep learning architectures, three CNN architectures are selected for building Dual CNN architectures. They are MobileNet (Light-weight), Xception (Separable convolutions with Residual connections) and DenseNet-121 (extract more number of features, feature reuse property) CNN architectures. The fused architecture is termed as Dual Deep Learning Architecture (DDLA). The DDLA architectures are used as a feature extractor. Later, the extracted features are classified using Machine Learning (ML) classifiers.

In this research work, the performance analysis of six Dual Deep Learning Architectures (DDLA) is carried out in the context of plant species recognition. The six DDLAs are categorized as similar DLs and dissimilar DLs.

The similar DL combinations are MobileNet + MobileNet (S1-DDLA), Xception + Xception (S2-DDLA) and DenseNet-121 + DenseNet-121 (S3-DDLA). The dissimilar DL combinations are MobileNet + Xception (DDLA-1), Xception + DenseNet-121 (DDLA-2) and MobileNet + DenseNet-121 (DDLA-3). The fused architectures are used as feature extractors. It is described in subsection 6.5.2. Then, the extracted features are classified by using the Machine Learning classifiers such as Linear Discriminant Analysis (LDA), multinomial Logistic Regression (LR), Naïve Bayes (NB), Classification and Regression Tree (CART), K-Nearest Neighbor (K-NN), Bagging Classifier (BC), Random Forest Classifier (RFC), Multi-layer Perceptron (MLP) and Support Vector Machine (SVM). Performance analysis

of Dual Deep Learning Architecture is carried out by using four datasets. Three datasets are benchmark datasets, namely, Flavia, Folio, and Swedish leaf dataset. Another dataset is a custom developed Leaf-12 dataset collected under various lighting conditions and color backgrounds.

6.1.1 Usage of Dual Deep Learning Architectures in Other Applications

Dual CNNs are used in applications, such as fast object detection (Pan *et al.* 2019), image segmentation (Du & Gao 2019), cancer segmentation (He *et al.* 2018), abnormal driving behavior, and speech emotion recognition (Ocquaye *et al.* 2019). Dual CNNs are also utilized in applications related to nature, such as smoke detection (Pundir & Raman 2019), aerial classification (Dede *et al.* 2018), and classification of land area (Minetto *et al.* 2019).

The organization of the chapter is as follows. The introduction section is followed by reporting the literature on plant species recognition using Dual CNN architectures (Section 6.2). In section 6.3, the proposed methodology for Dual Deep Learning Architecture is described. Experimental details and results are discussed in section 6.4. The summary of the chapter is provided in section 6.5.

6.2 LITERATURES RELATED TO DUAL DEEP LEARNING ARCHITECTURE FOR PLANT SPECIES RECOGNITION

In recent years, Dual architectures are widely used by different research groups across the world. The dual architectures are connected either in parallel or in a hierarchical manner. Some of the literatures using Dual Architectures for Plant Species Recognition is detailed below.

Rizk (2019) proposed a dual path CNN for plant species classification. The author utilized the leaves for classification. The author portrayed that the shape and venation features are vital for classification. The

author utilized the Sobel operators to extract the shape and venation information through patch generation. Dual-path CNN is used as a feature extractor. The author utilized the binarized image in the first path. The second path utilized leaf patches. Both the paths contain four 3x3 convolution layer followed by a ReLU and 2x2 max-pooling layer. The outputs from the two CNN paths are merged. It is followed by a Fully Connected Layer (FCL) with the Softmax activation function. It follows the early fusion scheme. Dual-path CNN is assessed on the augmented Flavia dataset. It resulted in a test accuracy of 96.8%. Augmented Flavia dataset contains about 3000 samples. This dual model is trained for 10 epochs using Adam optimizer with a learning rate of 0.01 and a batch size of 64.

Wang & Wang (2019) proposed a Siamese-Inception (S-Inception) CNN model. It is based on Siamese Neural Network and few-shot learning techniques. S-Inception incorporates two Inception-V4 models to extract the features from plant leaves. The authors proposed a Spatial Structure Optimizer (SSO) for metric formation using contrast loss function. The Euclidean distance metric is calculated for the test image and compared with the SSO metric. The similarity score is calculated using a K-NN classifier. The prediction with a maximum score is considered as an output label. The experiments are performed with a learning rate of 0.001, 30000 epochs, a dropout rate of 0.5, and a batch size of 8. The methodology is tested with Flavia, Swedish leaf, and LeafSnap datasets, resulting in accuracies of 95.32%, 91.67%, and 95.75%, respectively.

Li *et al.* (2019) proposed a Multi-branch CNN with Attention block (MCNNA) for plant species recognition. The attention block contains the upper and lower branches. Each branch has about nine convolution layers. In between the convolutional layers, two max-pooling layers and one deconvolution layer is used. The output feature map from the upper and lower branches are fused to form a feature map. The attention block focuses on the plant leaves and

branches. Multi-branch CNN contains three CNNs to extract the features of trunk, branch, and leaves. Attention block follows a multi-branch CNN. The feature map from the latter is combined to form a final feature vector. Multi-branch CNN consists of twelve convolution layer, four max-pooling layers, and two FCL. Each branch of multi-branch CNN utilizes different kernels and parameters. MCNNA method is trained using Stochastic Gradient Descent (SGD) with Momentum optimizer. The learning rate and batch size are set to 0.0005 and 50, respectively. The method is evaluated with BJFU100 and custom-developed datasets. The accuracies obtained for the datasets are 97.89% and 93.35%, respectively.

Zhu *et al.* (2019) proposed a Two-way Attention-based Deep Learning architecture (TA-CNN) for plant species recognition. TA-CNN is the combination of family first attention and max-sum attention. The first attention performs plant family classification for discriminative feature learning with the proposed implicit tree model. The Xception network with Global Average Pooling (GAP) is used for the first attention. The final few layers of the architecture are formed with two types of FCLs. The outputs of the FCLs are based on the number of families and the number of species. The max-sum attention is the second way used involving generation of the heat map, followed by max-sum part localization. The same Xception network is utilized in Fully Convolutional Neural Network for max-sum attention. This method is evaluated on four datasets, namely, Malayakew, ICL, Flowers102, and CFH. The data are augmented with eight scale rotation between 0° and 360°. The network is trained using an RMSprop optimizer with a learning rate and a decay rate of 0.0001 and 0.0007 per epoch.

Lee *et al.* (2018) proposed a Hybrid-Generic Convolutional Neural Network (HGO-CNN) method. HGO-CNN is accounted for the organ and generic information of plants. Also, the authors proposed a Plant-StructNet architecture based on Recurrent Neural Network (RNN), for multi-image

modeling and multi-plant views. The ensembling of HGO-CNN and Plant-StructNet model resulted in better performance. PlantCLEF 2015 dataset is used for evaluating the method. HGO-CNN is trained using the SGD optimizer with an initial learning rate of 0.01. The learning rate is decreased by a factor of 10 until the validation accuracy does not improve to a greater extent. Other parameters, such as momentum (0.9), weight decay (0.0001), and batch size (60), are also considered. The authors reported an accuracy value of 0.685, Borda count (BD) value of 0.747, and majority voting value (MAV) of 0.746.

He *et al.* (2018) presented a bi-channel framework using Deep Learning architectures for classifying the plant species. The bi-channel framework consists of VGG-16 and SqueezeNet CNN architectures. Both the models are trained with ImageNet weights and merged using a stacking layer. Then, the predictions from the individual models are fused to get the final prediction. The authors evaluated the method using 27 varieties of plants from the Orchidaceae family (Training-16455 images and Testing-1756 images). The reported dataset is a collection of web-crawled images from different search engines. SGD optimizer and log loss function are utilized in the training phase. A plant species recognition accuracy of 96.81% is reported.

Hu *et al.* (2018) presented a novel Multi-Scale Fusion Convolutional Neural Network (MSF-CNN) for plant species recognition. MSF-CNN network utilizes the images at different scales (downsampled using a bilinear interpolation method). MSF-CNN consists of nine Convolution-Batch Normalization-ReLU (CBR), five Max-Pooling (MP), three Concatenation (CAT), and one Average Pooling (AP) units. CBR unit contains a convolution layer, batch normalization, and ReLU activation function. MP represents a max-pooling operation (dimensionality reduction of feature maps). CAT represents the concatenation operation of the feature maps from different depths. After the third CAT unit, two sets of MP and CBR units, are added. Finally, the Softmax activation function is used with the AP unit. MSF-CNN

uses different image sizes (256x256, 128x128, 64x64 and 32x32) for processing. The parameters used in the experimental analysis are weight, bias, and regularization values. The weights are initialized using a Normal Distribution process. The bias values are initialized to zero. L2 regularization weight of 0.0001 is used. The other process parameters are batch size (128) and learning rate (0.01). MalayaKew and LeafSnap datasets are used to evaluate the MSF-CNN method.

Araujo *et al.* (2018) proposed a two-level (Genus and Species) plant classification method using a pre-trained CNN model. The dataset is augmented to increase the number of samples. Genus based plant classification is carried out by using two pre-trained GoogleNet CNN architectures. The predicted Genus of plant species is passed as an input to the two pre-trained GoogleNet CNN architectures. It is utilized for the classification of plant species. CNNs are trained using an SGD optimizer for 30 epochs (3432 iterations=1 epoch). The batch size and learning rate are set to 20 and 10^{-3}, respectively. The author's methodology minimizes the effect of intra-class variations. The method is tested using an ImageCLEF 2015 plant dataset. It resulted in an accuracy of 86.44%.

Lee *et al.* (2017) demonstrated the utilization of Dual CNN architectures in plant species recognition. Two CNNs are employed with two types of inputs, namely, leaf venation and patches. Raw pixel intensities are given as an input to the CNN. Important features are selected using Deconvolution Network (DN). The authors proposed three different methodologies for combining two CNN architectures. It is shown in figure 6.1. Early fusion methodology is represented in two ways, namely, Cascade and Conv-Sum. In the Cascaded Early Fusion method, the output feature map from the CNNs is concatenated before the softmax layer, as shown in figure 6.1(b). Figure 6.1(c) shows the conv-sum based Early Fusion method in which the feature maps are summed up prior to the softmax layer. Late fusion is obtained

by combining the prediction of individual CNNs. Averaging or majority voting scheme is employed for combining the prediction scores. The authors reported that the Early Fusion methodology performed better than the Late Fusion technique. An accuracy of 96.3% is obtained by utilizing the Early Fusion method based on conv-sum. The methodologies are tested using the Malayakew dataset.

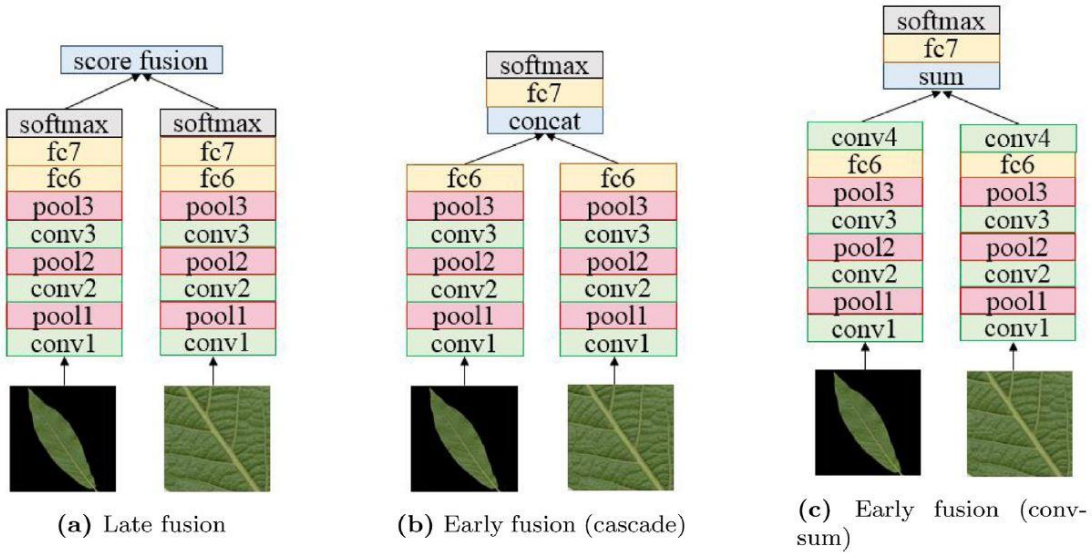


Figure 6.1 Hybrid fusion Methodologies (Source: Lee *et al.* 2017)

Ghazi *et al.* (2017) performed plant species identification using fine-tuned pre-trained Deep Learning models, namely, AlexNet, VGGNet, and GoogleNet CNN. Data augmentation operations (Translation, Rotation, Scaling, and Reflection) are used to minimize overfitting. The authors improved the prediction accuracy by fusing the fine-tuned CNN models (GoogleNet + VGGNet) with a sum rule to obtain the prediction scores. The authors reported an accuracy of 80% for the Dual CNN architecture using the LifeCLEF 2015 plant dataset.

Shah *et al.* (2017) proposed a dual path Deep Convolutional Neural Network for leaf classification. The method utilizes the shape and texture features. A leaf image and texture patch, are used as inputs to the dual-path

CNN. The first path contains a Batch Normalization (BN) process followed by seven sets of Convolution and ReLU activation function. Four Max-pooling layers are embedded in the first path. The second path consists of a BN followed by four convolution layers. Four Max-pooling layers are used in the second path. The dimensionally reduced output feature maps from the two paths are merged. Later, it is passed on to the two Fully Connected Layers (Number of neurons=4096 and number of output classes). The dual-path CNN is evaluated for three datasets, namely, Flavia, LeafSnap, and ImageCLEF.

Liu *et al.* (2015) proposed a hybrid deep learning method. It is formed by combining the AutoEncoder (AE) and CNN architecture. It is used in the feature extraction process. Further, the SVM classifier is used for classification. The configuration of AE is 4096-2048-1024-2048-4096, where each number represents the number of neurons/layer. The mid-layer (with 1024 neurons) output is fed as an input to the CNN architecture. The CNN consists of two convolution layers and one max pooling layer. The dimensionally reduced feature map from CNN architecture is used for classification by Support Vector Machine (SVM with Gaussian kernel). The authors reported that the performance of the hybrid method surpassed the individual methods (AE and CNN architecture). It resulted in an accuracy of 93% for the ICL leaf dataset.

Zhao *et al.* proposed a Growing Convolutional Neural Network (GCNN) for plant species recognition. In GCNN, the number of neurons/layer is increased for every additional convolution layer. The weight updation process continues until a minimum value for squared error is achieved. Initial GCNN architecture (Net-0) contains convolution (two numbers) and sampling layers. The network is trained with 367 parameters. A stopping criterion is also introduced. When the convergence speed is below the threshold, the first growth of architecture is performed. In the second round of network (Net-1), a new branch is initiated. Random weight assignment is carried out for the newly created neurons. Further, to improve the plant species recognition rate, a

Progressive Sample Learning Method (PSLM) is proposed. It is used to add more samples to the existing training set until the error drops below 0.3. The GCNN method is evaluated using the ImageCLEF2012 dataset. Its performance is better than traditional CNN and CNN with SVM classifier. The initialized process parameters are Learning rate (0.0002), Convergence Speed (threshold = 0.05) and Average Error (threshold = 0.2). An accuracy of 88.14% is obtained by using the GCNN (Nine Layers) with PSLM method.

Table 6.2 Summary of Literatures on Dual CNN Architectures for Plant Species Recognition

Method	Early Fusion/ Late Fusion/ Hierarchical	Transfer Learning (✓/×)	Data Augmentation (✓/×)	Author & Year
Dual-path CNN	Early Fusion	×	✓	Rizk (2019)
Siamese-Inception	Early Fusion	-	×	Wang & Wang (2019)
MCNNA	Early Fusion	×	×	Li *et al.* (2019)
TA-CNN	Hierarchical	✓	✓	Zhu *et al.* (2019)
HGO-CNN and Plant-StructNet Ensemble	Late Fusion	✓	✓	Lee *et al.* (2018)
Bi-channel Framework	Late Fusion	✓	×	He *et al.* (2018)
MSF-CNN	Hierarchical and early fusion	×	✓	Hu *et al.* (2018)
Hierarchical Classification (GoogleNet)	Hierarchical and late fusion	✓	✓	Araújo *et al.* (2018)
Hybrid global-local feature extraction (AlexNet)	Early Fusion (cascade, conv-sum) and Late Fusion	✓	✓	Lee *et al.* (2017)
GoogleNet + VGGNet	Late Fusion	✓	✓	Ghazi *et al.* (2017)
Dual-path deep CNN	Early Fusion	×	✓	Shah *et al.* (2017)
Hybrid deep learning	Hierarchical	×	×	Liu *et al.* (2015)
GCNN	Hierarchical	×	×	Zhao *et al.* (2014)

Table 6.2 summarizes the literatures on Dual CNN architectures for plant species recognition. It also provides details about the fusion methodology, data augmentation process, and transfer learning. Based on the literature survey, it is observed that the dual CNN architectures performed better compared to single CNN architectures. In some literatures, the usage of pre-trained weights and data augmentation process is reported. Also, it is observed that the architectures of similar length are combined to form Dual CNN architectures. The performance analysis of Dual CNN architectures in relation to computation time and real-time application is less studied.

Hence, in this research work, the individual CNN architectures are selected based on their characteristics to form Dual Deep Learning Architectures (DDLA) for plant species recognition. Three CNN architectures, namely, MobileNet, Xception, and DenseNet121 CNN architectures, are selected. MobileNet CNN is a lightweight architecture that requires less processing time. Xception CNN architecture utilizes the Separable convolutions with Residual connections. DenseNet-121 CNN architectures extract more number of features. Also, it supports the feature reuse property. The Dual Deep Learning Architectures are formed by combining the similar DL architectures and non-similar DL architectures. The DDLA is used as a feature extractor. Later, it is classified using the Machine Learning (ML) classifiers. The methodology is tested using four datasets, namely, Flavia, Folio, Swedish Leaf, and custom-developed Leaf-12 datasets.

6.3 DUAL DEEP LEARNING ARCHITECTURES FOR PLANT SPECIES RECOGNITION

The proposed Dual Deep Learning Architecture (DDLA) for plant species recognition is shown in figure 6.2. Plant images from the dataset are resized to 100x100 pixels. The normalization of pixel intensities is carried out, before forwarding the values to the Dual Architectures. Deep learning (DL)

CNN models act as a feature extractor. The output feature map (X1, X2) of DL models M1 and M2 are combined to form a final feature vector. Further, the classifier Cl is used to predict the label of plant species.

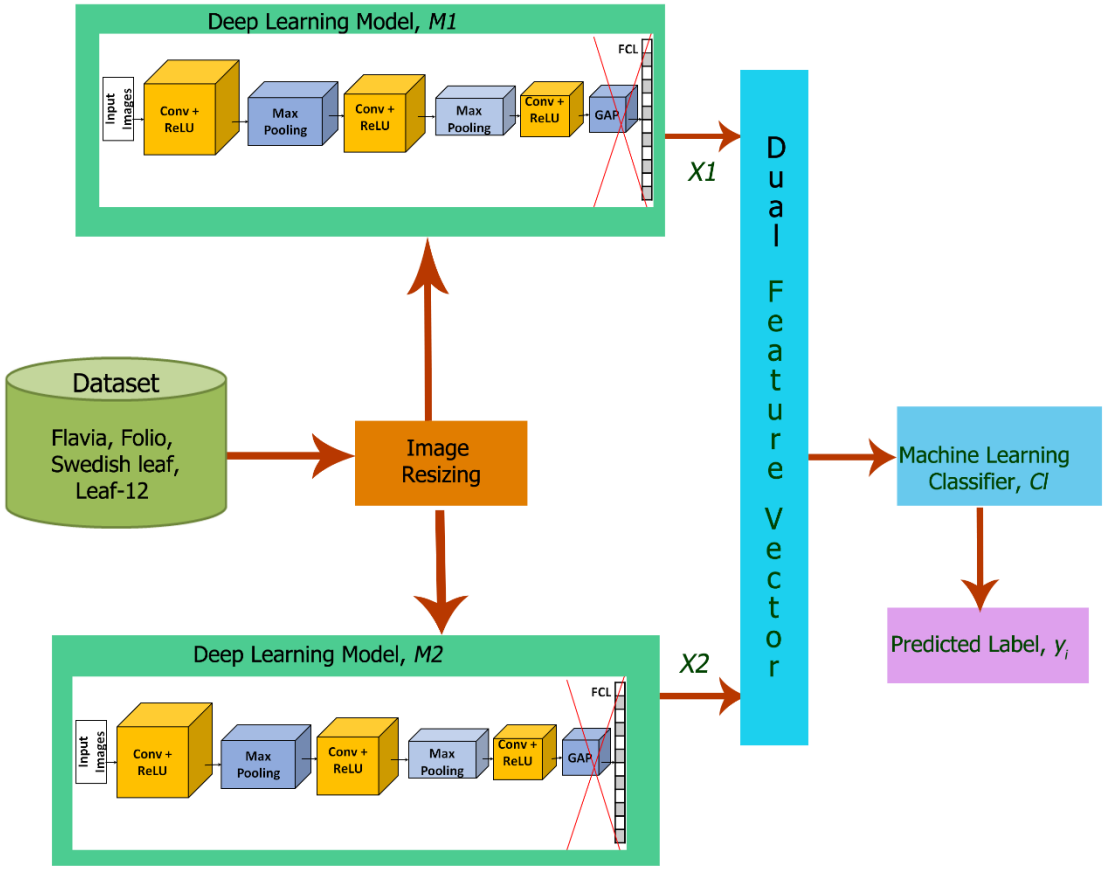


Figure 6.2 Proposed Dual Deep Learning Architecture for PSR

Steps involved in the process of plant species recognition system using DDLA is represented as an Algorithm 6.1. Four datasets, namely, Flavia, Folio, Swedish leaf, and Leaf-12, are used to evaluate the proposed methodology. ImageNet weights are used as initial weights for the Dual architectures.

Algorithm 6.1: Plant Species Recognition using DDLA

Input: Input Image, *I*; Models, *M1*, *M2*; Classifier, *Cl*; ImageNet weights, *IW* and biases, *IB*

Output: Feature Vector, X_1, X_2, X_{dual}; Predicted label, y_i

Step 1: Read the *N* number of images (*I*)

Step 2: Resize the images (*I*) to 100x100x3

Step 3: Process the individual CNN models (*M1* and *M2*) using *IW*, and *IB*. Take into account the number of images (*N*) in the dataset and the number of layers (*L*) in each CNN model.

Step 4: Feature Extraction

for i=1 to N

Obtain the feature vector, X1 from the CNN model, M1

Obtain the feature vector, X2 from the CNN model, M2

$$X_{dual} = concatenate(X1, X2)$$

Save 'X_{dual}' features in hdf5 file format

End for

Step 5: Apply classifier, *Cl* to the features, X_{dual} to predict the label, y_i

Step 6: Generate the classification report

6.3.1 Datasets and Image Resizing

Leaf datasets, namely, Flavia, Folio, Swedish leaf (Soderkvist 2002) and Leaf-12 are used. The total number of images per class in Flavia, Folio, Swedish leaf, and Leaf-12 datasets are 1600, 576, 1125, and 3840, respectively. All the images are resized to 300x300. Resizing is performed by maintaining its aspect ratio and image quality. Further, the images are resized to 100x100 using the nearest interpolation method.

6.3.2 Feature Extraction by Dual Architectures

Three Deep Learning architectures are chosen based on their characteristics to form the Dual Deep Learning Architectures (DDLA). The architectures are MobileNet CNN (lightweight network), Xception CNN (Separable convolutions with Residual connections) and DenseNet-121 CNN (Extracts more number of features and utilizes the feature reuse property).

Two types of dual architectures are taken into consideration, namely, similar and dissimilar DLs. Similar DLs include S1-DDLA, S2-DDLA, and S3-DDLA. S1-DDLA contains two MobileNet CNN architectures in parallel. S2-DDLA consists of two Xception CNN models arranged in a parallel fashion. S3-DDLA model contains two DenseNet CNN models in parallel branches. Dissimilar DLs include DDLA-1, DDLA-2, and DDLA-3. DDLA-1 is constructed using MobileNet and Xception CNN architectures. DDLA-2 consists of Xception and DenseNet-121 CNN architectures. MobileNet and DenseNet-121 CNN architectures form the DDLA-3. As discussed in chapter 5, the Global Average Pooling (GAP) and FCL in both the DL models are replaced with machine learning classifiers.

The reasons for selecting MobileNet, Xception, and DenseNet-121 architectures are based on the properties of the architectures. MobileNet is a lightweight CNN model (Howard *et al.* 2017); Xception contains the residual connections as in ResNet CNN, 3x3 convolutions as in Inception-V3 and Separable convolutions as in MobileNet CNN; DenseNet-121 CNN contains 121 layers being the minimum among its variants (DenseNet-169, DenseNet-201). The performance of DenseNet-121 CNN is better than the ResNet CNN model.

6.3.2.1 S1-DDLA (MobileNet + MobileNet CNN)

In S1-DDLA, two MobileNet CNN architectures are processed in parallel. The block diagram representation for S1-DDLA is shown in figure 6.3. MobileNet CNN is a lightweight architecture utilizing the Depthwise Separable Convolutions (DWSC). A feature vector of 9216 dimensions is obtained from each MobileNet CNN model. The feature vectors obtained by individual MobileNet CNN architectures are represented by $X1_{MOB}$ and $X2_{MOB}$. The feature vectors are concatenated to form a 18432-d feature vector (X_{S1}), which is represented in Equation (6.1).

$$X_{S1} \leftarrow concatenate(X1_{MOB}, X2_{MOB}) \quad (6.1)$$

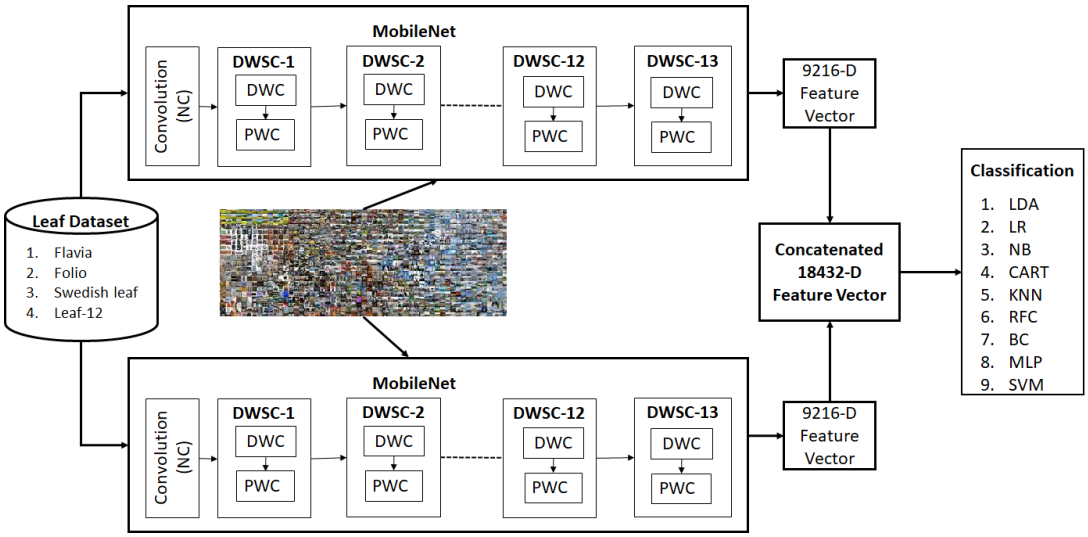


Figure 6.3 S1-DDLA (MobileNet + MobileNet CNN) for PSR

6.3.2.2 S2-DDLA (Xception + Xception CNN)

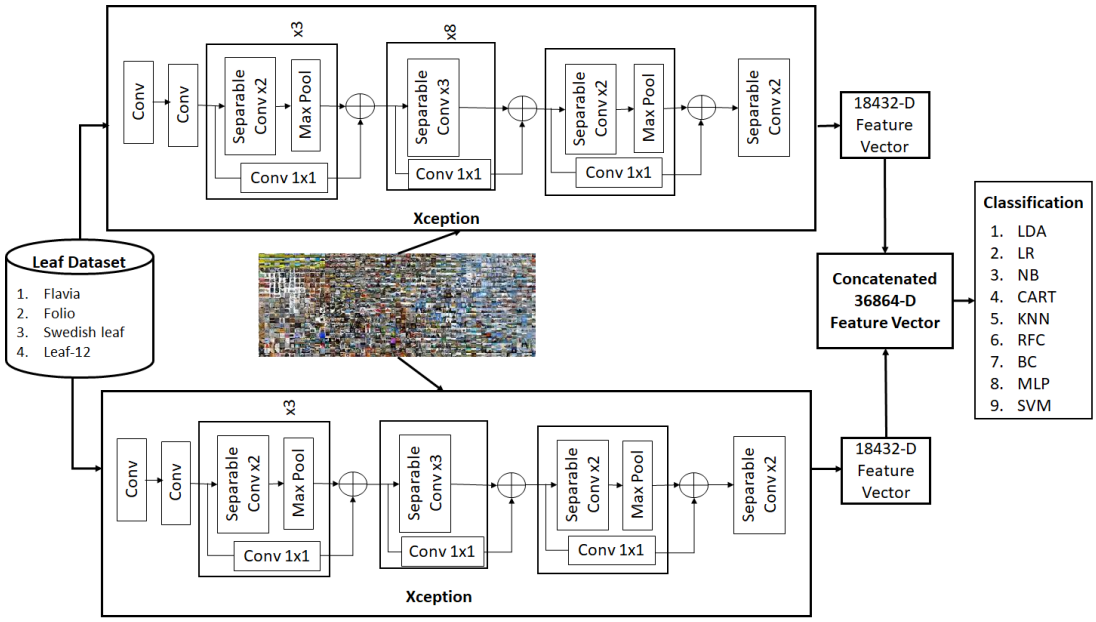


Figure 6.4 S2-DDLA (Xception + Xception CNN) for PSR

S2-DDLA is presented in figure 6.4. It contains two Xception CNN architectures connected in parallel. The Xception CNN model consists of convolution and separable convolution layers. Residual connections are present between the convolutions. Some of the residual connections hold a 1x1

convolution. GAP, FCL, and LR classifier used in the original Xception model are not considered for this CNN model. A 1-D feature vector of dimension 18432 is obtained from the last separable convolution layer of the Xception CNN model. Equation (6.2) represents the concatenation of features, $X1_{XCEP,}$ and $X2_{XCEP}$ from two Xception CNNs.

$$X_{S2} \leftarrow concatenate(X1_{XCEP}, X2_{XCEP}) \qquad (6.2)$$

6.3.2.3 S3-DDLA (DenseNet-121 + DenseNet-121 CNN)

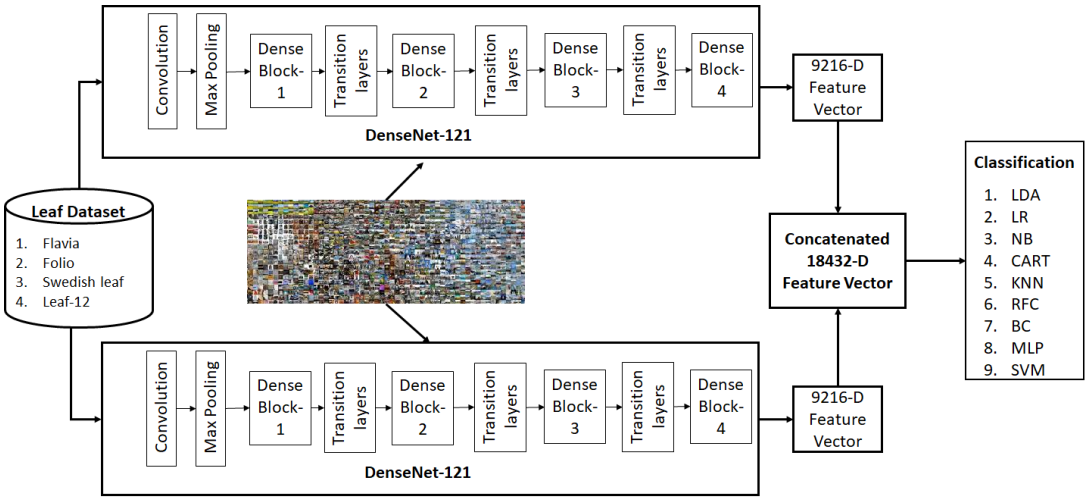


Figure 6.5 S3-DDLA (DenseNet-121 + DenseNet-121 CNN) for PSR

The parallel arrangement of two DenseNet-121 CNN architectures forms the S3-DDLA. Figure 6.5 shows the block diagram representation of S3-DDLA. DenseNet-121 CNN architecture minimizes the problem of Vanishing Gradient through the feature reuse property. It includes a convolution layer and a max-pooling layer followed by four dense blocks. The transition layer is sandwiched between the dense blocks. A feature vector of 9216 dimensions is obtained from the dense block-4 of each DenseNet-121 CNN architecture. The features $X1_{DEN121}$, $X2_{DEN121}$ obtained from the two DenseNet-121 CNN architectures are concatenated to form a 18432 dimension feature vector (X_{S3}). It is represented in Equation (6.3).

$$X_{S3} \leftarrow concatenate(X1_{DEN121}, X2_{DEN121}) \qquad (6.3)$$

6.3.2.4 DDLA-1 (MobileNet + Xception CNN)

The block diagram representation of DDLA-1 is shown in figure 6.6. It consists of two deep learning models connected in parallel. The CNN models are MobileNet CNN and Xception CNN architectures. MobileNet CNN and Xception CNN architectures contain convolution layers and separable convolution layers. Residual connections (hold a 1 x 1 convolutions) are part of Xception CNN architecture. GAP, FCL and LR classifier, are removed from the original Xception CNN architecture.

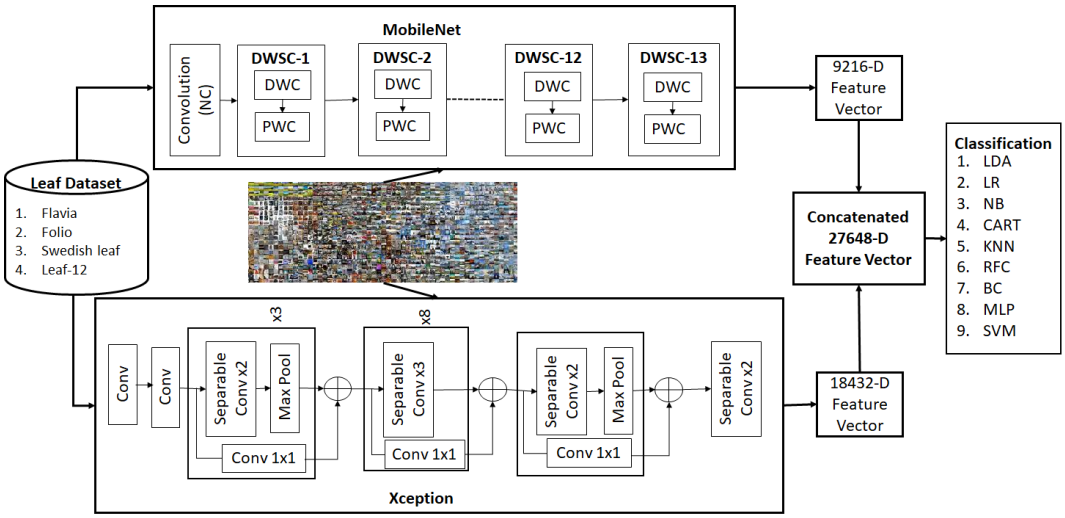


Figure 6.6 DDLA-1 (MobileNet + Xception CNN) for PSR

A feature vector, $X1_{MOB}$ of 9216 dimensions, is obtained using MobileNet. A 1-D feature vector ($X2_{XCEP}$) of 18432 dimensions is obtained from the last separable convolution layer of the Xception CNN model. The features $X1_{MOB}$ and $X2_{XCEP}$ are concatenated to form a final feature vector, X_{DDLA1} having 27,648 dimensions. It is represented in Equation (6.4).

$$X_{DDLA1} \leftarrow concatenate(X1_{MOB}, X2_{XCEP}) \qquad (6.4)$$

6.3.2.5 DDLA-2 (Xception + DenseNet-121 CNN)

Figure 6.7 shows the block diagram representation of DDLA-2. It consists of two CNN models, namely, Xception CNN and DenseNet-121 CNN

architectures. The two architectures are connected in parallel. Xception CNN architecture contains convolutions and separable convolutions layers. Residual connections (hold a 1x1 convolution) are also a part of Xception CNN architecture.

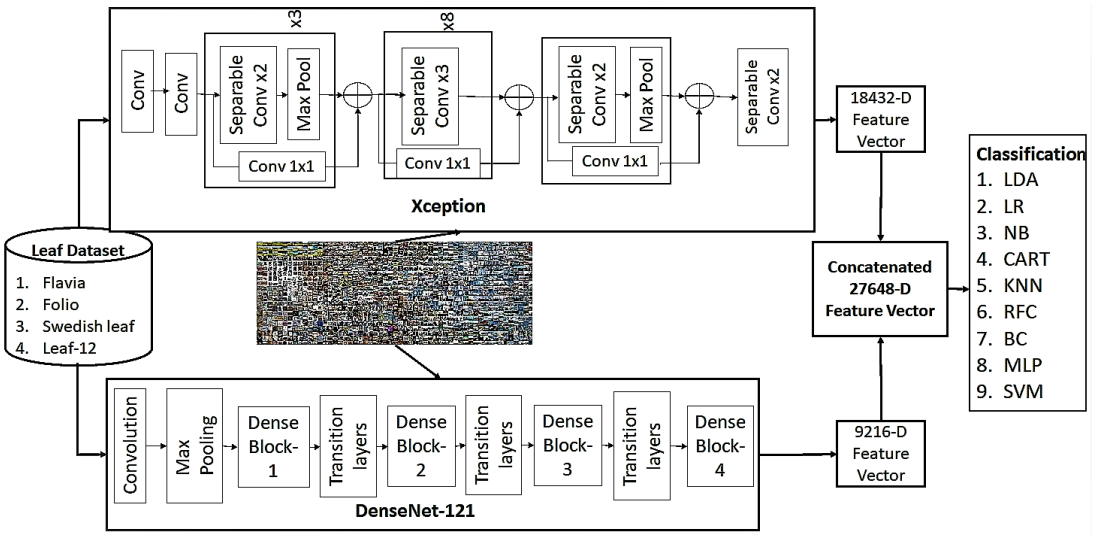


Figure 6.7 DDLA-2 (Xception + DenseNet-121 CNN) for PSR

Global Average Pooling (GAP), Fully Connected Layer (FCL), and Logistic Regression (LR) classifier, from the Xception CNN model, are removed. These layers are replaced by Machine Learning classifiers. A 1-D feature vector ($X1_{XCEP}$) of 18432-dimension is obtained from the last separable convolution layer of the Xception CNN model.

A feature vector ($X2_{DEN}$) of 9216-dimension is obtained from the DenseNet-121 architecture. The features from the Xception and DenseNet-121 CNN architectures are concatenated to form a 27648-dimension feature vector (X_{DDLA2}). It is represented in Equation (6.5).

$$X_{DDLA2} \leftarrow concatenate(X1_{XCEP}, X2_{DEN}) \qquad (6.5)$$

6.3.2.6 DDLA-3 (MobileNet + DenseNet-121 CNN)

The block diagram representation of DDLA-3 is shown in figure 6.8. It consists of two CNN architectures. They are MobileNet CNN and DenseNet-

121 CNN architectures. The Dual architectures are connected in parallel. MobileNet CNN architecture utilizes the Normal Convolution and DepthWise Separable Convolution (DWSC) process. DepthWise Separable Convolution process consists of DepthWise convolution (DWC) and PointWise Convolutions (PWC). On processing the 13th Depthwise Separable Convolution layer (DWSC), a feature vector ($X1_{MOB}$) of 9216-dimension is obtained.

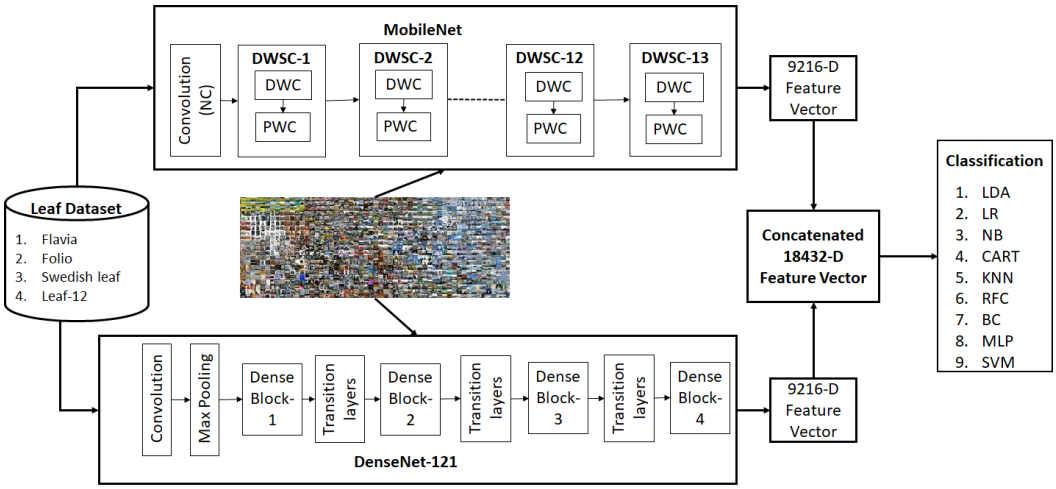


Figure 6.8 DDLA-3 (MobileNet + DenseNet-121 CNN) for PSR

The DenseNet-121 CNN model includes the convolution layer, max-pooling layer, followed by four dense blocks. The transition block is included between the dense blocks. The output feature map from the current layer is concatenated with the outputs of all the previous layers. A feature vector of 9216 dimensions is obtained from dense block-4 of the DenseNet-121 CNN model. The features from the MobileNet ($X1_{MOB}$) and DenseNet-121($X2_{DEN}$) are concatenated to form a 18432 dimension feature vector (X_{DDLA3}). It is represented in Equation (6.6).

$$X_{DDLA3} \leftarrow concatenate(X1_{MOB}, X2_{DEN}) \qquad (6.6)$$

6.3.3 Plant Species Classification by Machine Learning Classifiers

The extracted features from the Dual Deep Learning Architectures (DDLA) are classified using Machine Learning (ML) classifiers. The ML classifiers used are Multinomial Logistic Regression (LR), Linear Discriminant Analysis (LDA), Naïve Bayes (NB), K-Nearest Neighbor (K-NN), Classification and Regression Tree (CART), Random Forest Classifier (RFC), Bagging Classifier (BC), Multi-Layer Perceptron (MLP) and Support Vector Classifier (SVM) (Marsland 2014; Mitchell 1997).

The fine-tuned DDLA (CNN as feature extractor and classifier) resulted in high loss values, leading to poor accuracy. Also, the computational time is high and suffers overfitting when data augmentation is performed. Hence, the results related to the fine-tuned DDLA system for plant species recognition is not included. In this research work, the DDLA system is used to extract the features from the plant leaf images. Later, it is classified using Machine Learning classifiers.

6.4 RESULTS AND DISCUSSION

The performance analysis of Dual Deep Learning Architectures (Similar DLs and Dissimilar DLs) with Machine Learning (ML) classifiers is carried out. The Dual Deep Learning Architectures (DDLA) are used as feature extractors. Similar DLs are S1-DDLA (MobileNet CNN + MobileNet CNN), S2-DDLA (Xception CNN + Xception CNN) and S3-DDLA (DenseNet-121 CNN + DenseNet-121 CNN). Dissimilar DLs are DDLA-1 (MobileNet CNN + Xception CNN), DDLA-2 (Xception CNN + DenseNet-121 CNN) and DDLA-3 (MobileNet CNN + DenseNet-121 CNN). Then, the extracted features are classified using ML classifiers. The classifiers used are Linear Discriminant Analysis (LDA), multinomial Logistic Regression (LR), Naive Bayes (NB), K-Nearest Neighbor (K-NN), Classification and Regression Tree (CART),

Random Forest (RF), Bagging Classifier (BC), Multi-Layer Perceptron (MLP) and Support Vector Machine (SVM).

The images from the plant datasets are segregated into training and testing set in the ratio of 70:30. All the DDLAs are implemented using Windows 10 64-bit OS, i7 processor, and NVIDIA Titan X GPU. Python framework with supporting packages is used for software implementation. The supporting packages are OpenCV, Numpy, OS, Seaborn, Matplotlib, Scikit-Learn, Keras (Tensorflow Backend), and H5py.

6.4.1 Flavia Dataset

The results obtained by using Flavia dataset are listed in Table 6.3. From the table, it is observed that the DDLA-3 architecture (MobileNet CNN + DenseNet-121 CNN) resulted in higher accuracy compared to other combinations of architectures. The accuracy of DDLA-3 is 98.71%. It is also observed that the multinomial Logistic Regression classifier performed better compared to other classifiers, irrespective of the Dual Deep Learning Architectures.

Table 6.3 Accuracy values obtained for DDLAs by varying ML Classifiers

DDLA	Machine Learning Classifiers								
	LDA	**LR**	**NB**	**KNN**	**CART**	**RFC**	**BC**	**MLP**	**SVM**
S1-DDLA	60.00	**96.67**	93.12	95.00	69.58	95.42	88.54	94.79	96.25
S2-DDLA	81.25	**92.50**	64.38	82.92	57.71	86.04	73.96	90.83	91.25
S3-DDLA	87.08	**97.92**	83.54	93.96	73.54	96.25	88.33	96.88	96.67
DDLA-1	83.12	**96.88**	93.12	94.58	70.00	94.38	87.92	95.42	96.04
DDLA-2	94.79	**98.12**	85.42	94.79	70.42	96.04	89.79	96.37	97.29
DDLA-3	89.38	**98.71**	92.50	95.83	74.79	97.08	87.92	96.67	96.45

The obtained result for DDLA-3 (MobileNet CNN + DenseNet-121 CNN) + LR classifier is compared with existing literatures for the Flavia dataset. It is represented in Table 6.4. The computational time (Time taken for Training and Testing) is also estimated and tabulated in Table 6.4. It is observed that the proposed method (DDLA-3 + LR classifier) performed better compared to other existing literatures on Dual Architectures, in terms of accuracy (98.71%) and computation time (Training Time = 239.27 s). It is also observed that the Dual Deep Learning Architecture (DDLA) with Machine Learning classifier (LR) resulted in higher accuracy compared to other approaches (Conventional Method: CSTF+LDA, Neural Network: P-BPNN, Single DL Architecture: fine-tuned DenseNet-169 CNN).

The confusion matrix obtained by using DDLA-3 +LR classifier for plant species recognition is shown in figure 6.9. It is observed that the number of mispredictions is significantly reduced using DDLA-3 with LR classifier.

Table 6.4 Comparison of Existing literatures with Proposed DDLA Architecture

Method	**Accuracy (%)**	**Computational time (in s)**	
		Train Time	**Test Time**
PReLU based Backpropagation Neural Network (P-BPNN)	85.83	24.15	0.06
Color, Shape Texture Fusion (CSTF) + Linear Discriminant Analysis (LDA)	89.17	85.41	0.03
S-Inception (Wang & Wang 2019)	95.32	-	-
Dual-path CNN (Rizk 2019)	96.8	-	-
Fine-tuned DenseNet-169 CNN	98.58	422.82	3.53
DDLA-3 + Logistic Regression (LR)	**98.71**	**239.27**	**0.02**

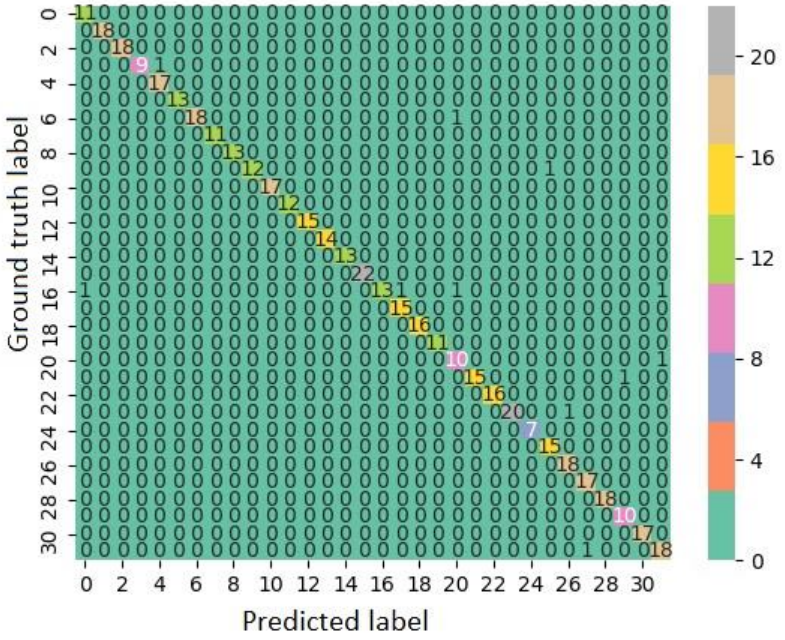


Figure 6.9 Confusion matrix obtained for DDLA-3+LR- Flavia dataset

6.4.2 Folio Dataset

Table 6.5 Accuracy values obtained for DDLAs by varying ML Classifiers

DDLA	Machine Learning Classifiers								
	LDA	**LR**	**NB**	**KNN**	**CART**	**RFC**	**BC**	**MLP**	**SVM**
S1-DDLA	86.71	**93.06**	88.44	89.60	54.91	89.60	75.14	88.44	91.33
S2-DDLA	83.24	**87.86**	70.52	77.46	48.55	78.03	63.58	83.82	84.97
S3-DDLA	91.91	**95.95**	85.55	89.60	60.69	91.33	80.35	94.22	93.06
DDLA-1	80.92	**93.64**	89.60	91.33	60.12	87.86	71.68	88.44	93.54
DDLA-2	90.17	**95.38**	87.86	90.75	64.74	91.33	81.45	94.02	94.22
DDLA-3	90.75	**96.38**	90.75	93.06	58.96	92.49	81.50	91.91	94.80

The results obtained by using the Folio dataset are listed in Table 6.5. From the table, it is observed that DDLA-3 architecture (MobileNet CNN +

DenseNet-121 CNN) resulted in higher accuracy compared to other combinations of architectures. DDLA-3 resulted in an accuracy of 96.38%. It is also observed that the multinomial Logistic Regression classifier performed better compared to other classifiers, irrespective of the Dual Deep Learning Architectures.

The obtained result for DDLA-3 (MobileNet CNN + DenseNet-121 CNN) + LR classifier is compared with other methodologies for the Folio dataset. It is represented in Table 6.6. The computation time (Time taken for Training and Testing process) is also estimated and tabulated in Table 6.6. From the table content, it is observed that the Dual Deep Learning Architecture (DDLA-3) with Machine Learning classifier (LR) resulted in higher accuracy (96.38%) compared to other approaches (Conventional Method: CSTF+LDA, Neural Network: P-BPNN, Single DL Architecture: DenseNet-121 CNN + LR, DenseNet-169 CNN + MLP, fine-tuned DenseNet-169 CNN, fine-tuned DenseNet-201 CNN). There are no existing literatures that use Dual Architecture on Folio dataset.

Table 6.6 Comparison of Other Methodologies with Proposed DDLA Architecture

Method	**Accuracy (%)**	**Computational time (in s)**	
		Train Time	**Test Time**
PReLU based Backpropagation Neural Network (P-BPNN)	76.88	9.63	0.05
Color, Shape, Texture Fusion (CSTF) + Linear Discriminant Analysis (LDA)	79.77	33.89	0.01
DenseNet-121+Logistic Regression (LR)	95.95	52.76	0.01
DenseNet-169+ Multi-Layer Perceptron (MLP)	95.95	209.81	0.10
Fine-tuned DenseNet-169 CNN	95.95	188.29	3.12
Fine-tuned DenseNet-201 CNN	95.95	227.33	3.96
DDLA-3+LR	**96.38**	**60.82**	**0.01**

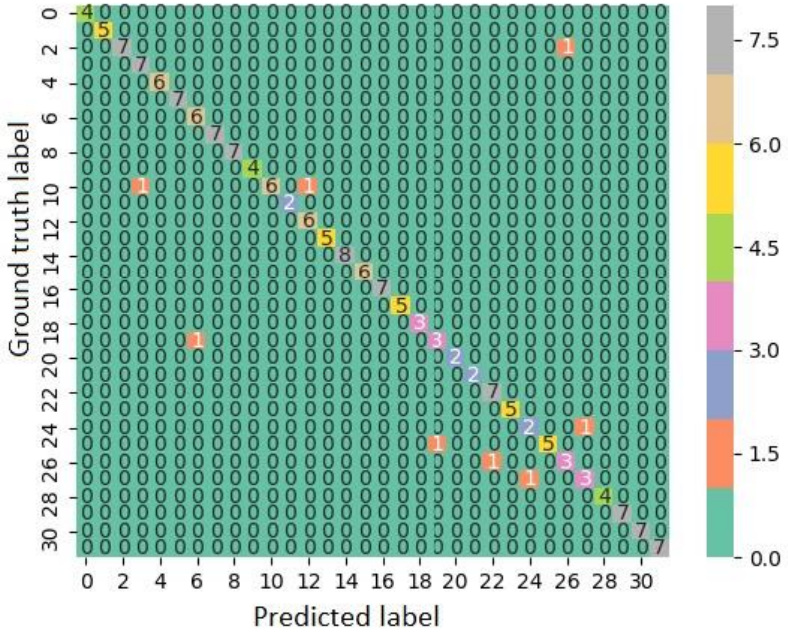


Figure 6.10 Confusion Matrix obtained by using DDLA-3 + LR – Folio Dataset

The confusion matrix obtained by using DDLA-3 +LR classifier for plant species recognition is shown in figure 6.10. It is observed that the number of mispredictions is significantly reduced using DDLA-3 with LR classifier.

6.4.3 Swedish leaf Dataset

Table 6.7 Accuracy values obtained for DDLAs by varying ML Classifiers

DDLA	Machine Learning Classifiers								
	LDA	**LR**	**NB**	**KNN**	**CART**	**RFC**	**BC**	**MLP**	**SVM**
S1-DDLA	94.67	**99.13**	91.72	95.86	80.18	95.86	93.49	98.22	99.41
S2-DDLA	90.53	**96.75**	79.88	89.35	74.56	92.01	86.98	94.38	96.15
S3-DDLA	93.20	**99.23**	85.21	94.97	79.29	97.93	93.79	97.63	98.22
DDLA-1	96.15	**99.00**	92.60	96.45	79.29	96.75	94.38	96.75	99.11
DDLA-2	98.52	**98.82**	89.35	96.15	82.25	97.63	94.67	97.34	98.82
DDLA-3	97.34	**99.41**	95.27	97.04	82.25	99.11	95.86	98.22	99.41

The results obtained for the Swedish Leaf dataset are listed in Table 6.7. From the table, it is observed that DDLA-3 architecture (MobileNet CNN + DenseNet-121 CNN) resulted in higher accuracy compared to other combinations of architectures. DDLA-3 resulted in an accuracy of 99.41%. Also, it is observed that the multinomial Logistic Regression classifier performed better compared to other classifiers, irrespective of the Dual Deep Learning Architectures.

Table 6.8 Comparison of Existing Literatures with Proposed DDLA Architecture

Method	Accuracy (%)	Computational time (in s)	
		Train Time	Test Time
S-Inception (Wang & Wang 2019)	91.67	-	-
Haralick Texture (HT) + Hu Moments (HM)+ Linear Discriminant Analysis (LDA)	92.01	54.88	0.02
PReLU based Backpropagation Neural Network (P-BPNN)	94.97	23.28	0.06
MobileNet+ Logistic Regression (LR)	99.41	66.5	0.01
Fine-tuned DenseNet-121 CNN	99.41	296.29	2.43
Fine-tuned DenseNet-169 CNN	99.41	313.13	3.41
DDLA-3+LR	**99.41**	**80.62**	**0.02**

The obtained result for DDLA-3 (MobileNet CNN + DenseNet-121 CNN) + LR classifier is compared with existing literatures for Swedish Leaf dataset. It is represented in Table 6.8. The computation time (Time taken for Training and Testing) is also estimated and tabulated in Table 6.8. From the table content, it is observed that the proposed method (DDLA-3 + LR classifier) performed better compared to other existing literatures on Dual Architectures, in terms of accuracy (99.41%) and computation time (Training Time = 80.62 s). Also, it is observed that the Dual Deep Learning Architecture (DDLA) with Machine Learning classifier (LR) resulted in higher accuracy compared to other

approaches (Conventional Method: Haralick Texture and Hu Moments + Linear Discriminant Analysis (LDA), Neural Network: P-BPNN). Single Deep Learning architecture (MobileNet CNN + LR, fine-tuned DenseNet-121 CNN, fine-tuned DenseNet-169 CNN) resulted in a similar accuracy (99.41%) to that of DDLA-3 (MobileNet + DenseNet-121 CNN) with LR classifier. It is also observed that the single Deep Learning architecture (MobileNet CNN+LR) produced the result at lower computation time (Training Time: 66.50 s).

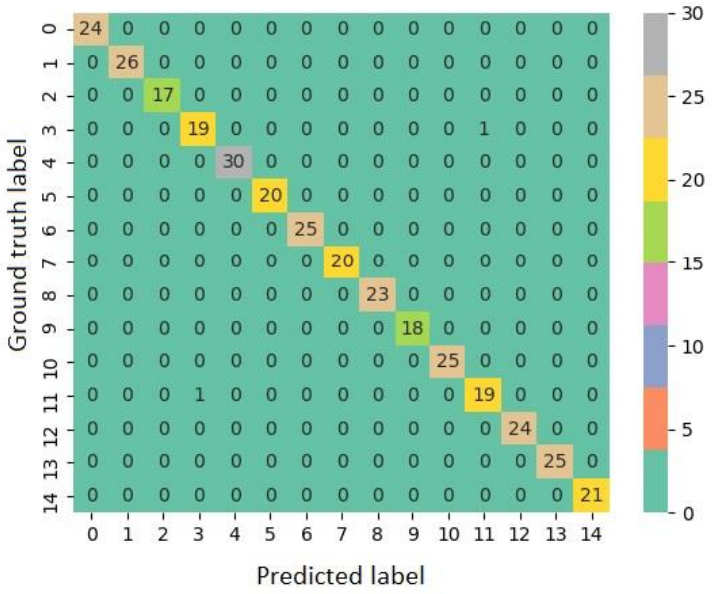


Figure 6.11 Confusion Matrix obtained for DDLA-3 + LR- Swedish leaf dataset

The confusion matrix obtained by using DDLA-3 +LR classifier for plant species recognition is shown in figure 6.11. It is observed that the number of mispredictions is significantly reduced using DDLA-3 with LR classifier.

6.4.4 Leaf-12 Dataset

The results obtained for the Leaf-12 dataset are listed in Table 6.9. It is observed that DDLA-3 architecture (MobileNet CNN + DenseNet-121 CNN) resulted in higher accuracy compared to other combinations of architectures. DDLA-3 resulted in an accuracy of 99.39%. Also, it is observed that the

multinomial Logistic Regression classifier performed better compared to other classifiers, irrespective of the Dual Deep Learning Architectures.

Table 6.9 Accuracy values obtained for DDLAs by varying ML Classifiers

DDLA	Machine Learning Classifiers								
	LDA	**LR**	**NB**	**KNN**	**CART**	**RFC**	**BC**	**MLP**	**SVM**
S1-DDLA	96.09	**97.48**	59.11	94.01	61.20	93.23	80.30	96.70	97.22
S2-DDLA	92.45	**94.53**	66.93	87.76	56.08	85.94	72.14	93.84	94.36
S3-DDLA	98.00	**98.09**	88.11	95.23	68.32	95.49	84.55	97.74	97.83
DDLA-1	96.96	**97.83**	59.20	94.62	61.20	92.36	79.25	97.14	97.74
DDLA-2	97.57	**98.18**	81.08	95.57	67.62	95.05	86.20	97.92	97.82
DDLA-3	99.05	**99.39**	57.99	96.27	69.18	96.53	85.85	98.78	99.31

The obtained result for DDLA-3 (MobileNet CNN + DenseNet-121 CNN) + LR classifier is compared with other methods (Conventional method, Neural Network, Single Deep Learning Architecture) using Leaf-12 dataset. It is represented as Table 6.10.

Table 6.10 Comparison of Other Methodologies with Proposed DDLA Architecture

Method	**Accuracy (%)**	**Computational time (in s)**	
		Train Time	**Test Time**
Color, Shape, Texture Fusion (CSTF) + Random Forest Classifier (RFC)	82.38	205.39	11.36
PReLU-based Backpropagation Neural Network (P-BPNN)	87.50	18.72	0.1
Fine-tuned DenseNet-169 CNN	99.00	966.02	4.45
DDLA-3+LR	**99.39**	**541.45**	**0.04**

The computation time (Time taken for Training and Testing process) is also estimated and tabulated in Table 6.10. It is observed that the Dual Deep Learning Architecture (DDLA) with Machine Learning classifier (LR) resulted in higher accuracy (99.39%) compared to other approaches (Conventional Method: CSTF+RFC, Neural Network: P-BPNN, Single DL Architecture: fine-tuned DenseNet-169 CNN). It is also observed that the higher accuracy is obtained at lower computation time (Training Time: 541.45 s).

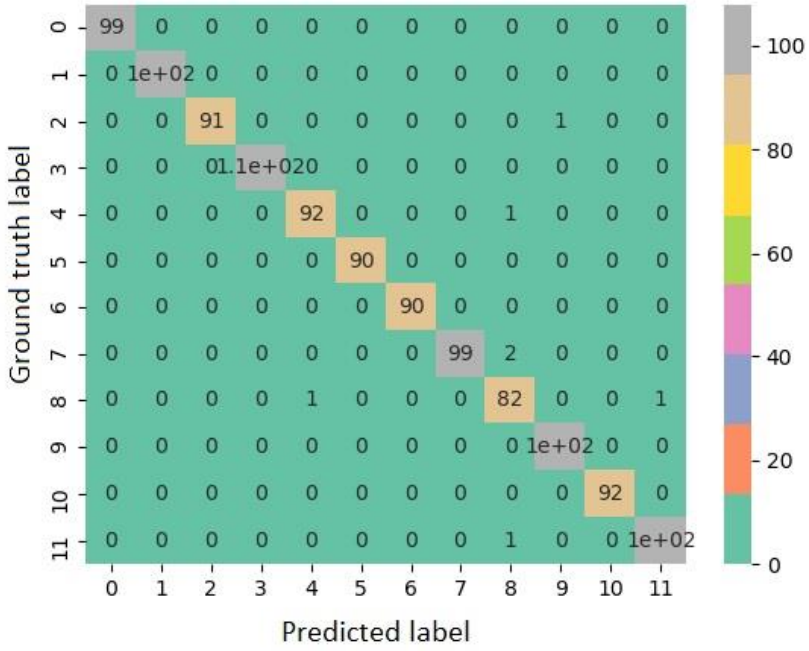


Figure 6.12 Confusion Matrix obtained for DDLA-3 + LR- Leaf-12 dataset

The confusion matrix obtained by using DDLA-3 +LR classifier for plant species recognition is shown in figure 6.12. It is observed that by using DDLA-3 with LR classifier, the number of mispredictions is substantially reduced.

6.4.4.1 Comparison of results obtained by DDLA-3 (MobileNet CNN + DenseNet-121 +LR classifier) with Dual Architecture (MobileNet CNN + DenseNet-169 CNN + ML classifier)

For Leaf-12 dataset, a single Deep Learning architecture (fine-tuned DenseNet-169 CNN) resulted in higher accuracy of about 99%. Hence, in Dual Deep Learning Architecture, the combination of MobileNet CNN with DenseNet-169 CNN is evaluated for plant species recognition. The comparison of results obtained by DDLA-3 (MobileNet CNN + DenseNet-121) with Dual Architecture (MobileNet CNN + DenseNet-169 CNN) is plotted as a graph in figure 6.13. It is observed that the DDLA-3 (MobileNet CNN + DenseNet-121) + LR classifier resulted in higher accuracy compared to Dual Architecture with a combination of MobileNet CNN + DenseNet-169 CNN + LR classifier.

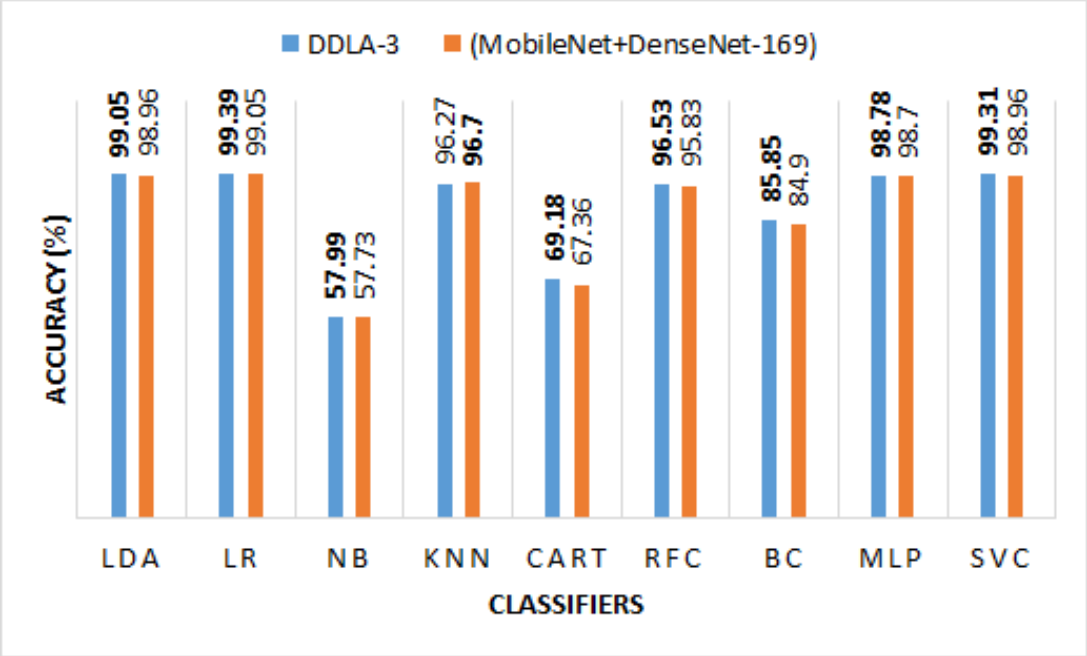


Figure 6.13 Comparison of results obtained by DDLA-3 and MobileNet + DenseNet-169 Architectures

6.4.4.2 Real-time Prediction of leaf using DDLA-3 + LR

Single Deep Learning architectures resulted in more number of mispredictions when it is exposed to the real-time dataset. The Dual Deep Learning Architectures (DDLA) with Machine Learning classifiers is proposed to minimize the number of mispredictions. Based on the experimental investigation, it is observed that the DDLA-3 (MobileNet CNN + DenseNet-121 CNN + LR classifier) performed better compared to other combinations of architectures. Hence, the real-time leaf images are tested with the DDLA-3 + LR classifier. The results are shown in figures 6.14 to 6.17. The results obtained by using four plant species (Indian Borage, Jungle Flame, Rose, and Turkey Berry) are reported in this thesis.

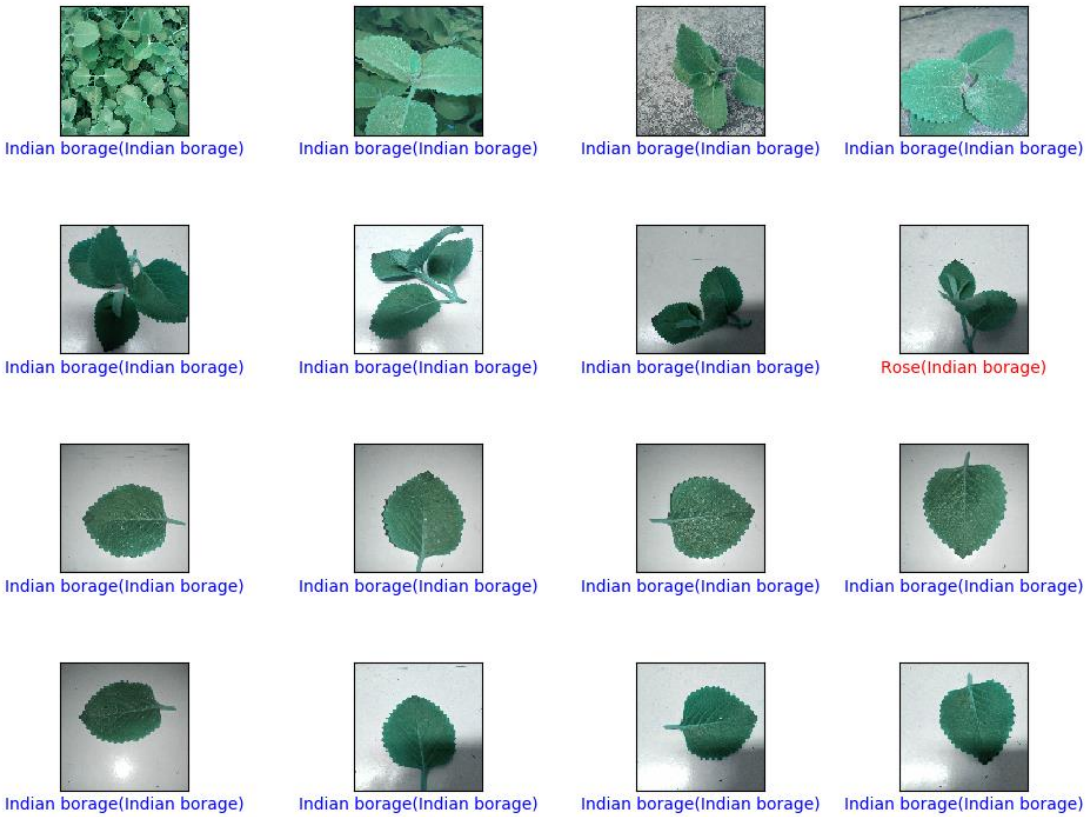


Figure 6.14 Real-time leaf (Indian borage) Prediction using DDLA-3 + LR classifier

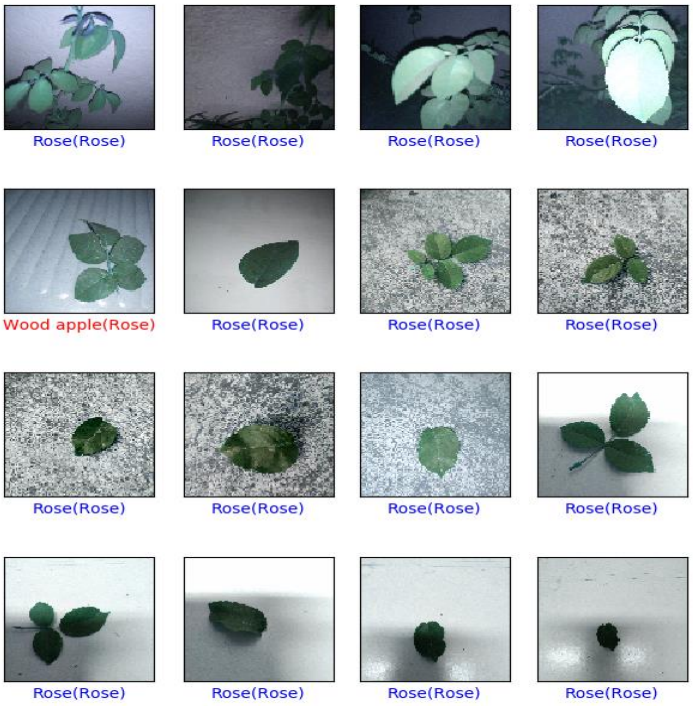


Figure 6.15 Real-time leaf (Rose) Prediction using DDLA-3+LR classifier

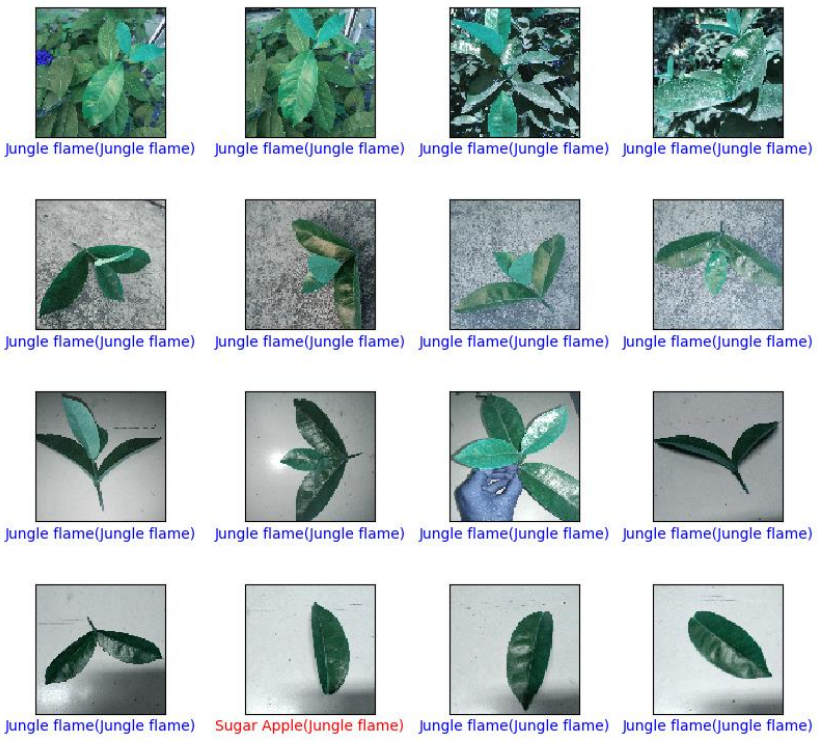


Figure 6.16 Real-time leaf (Jungle Flame) Prediction using DDLA-3 + LR classifier

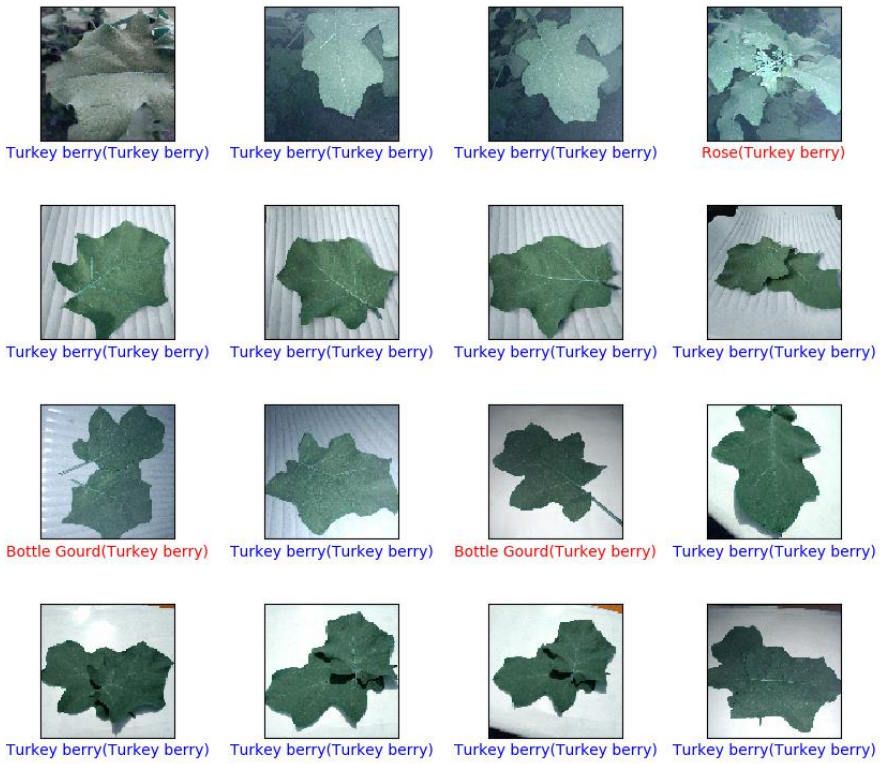


Figure 6.17 Real-time leaf (Turkey berry) Prediction using DDLA-3 + LR classifier

The real-time plant images are obtained by varying the scale (distance between the leaf to camera = 10 cm to 30 cm), orientation, camera viewpoints, illumination, and leaf (simple and compound leaves). In results (figure 6.14 to 6.17), a label is used to indicate the predicted value and ground truth value. The ground truth values are enclosed in brackets. It is observed that the number of mispredictions is significantly reduced by using the DDLA-3 with LR classifier.

6.5 SUMMARY

The performance analysis of Dual Deep Learning Architectures (DDLA) is carried out for plant species recognition. The DDLAs are used as feature extractors. The extracted features are classified using Machine Learning (ML) classifiers. This chapter addressed two DDLA approaches, namely, Similar DLs and Dissimilar DLs. In similar DLs, three combinations are

considered. They are S1-DDLA (MobileNet CNN + MobileNet CNN), S2-DDLA (Xception CNN + Xception CNN) and S3-DDLA (DenseNet-121 CNN + DenseNet-121 CNN). In dissimilar DLs, three combinations are considered. They are DDLA-1 (MobileNet CNN + Xception CNN), DDLA-2 (Xception CNN + DenseNet-121 CNN) and DDLA-3 (MobileNet CNN + DenseNet-121 CNN).

Table 6.11 Performance Metrics obtained by using DDLA+LR Classifier (Similar DL and Dissimilar DL)

Dataset	CNN Model	Accuracy (%)		Performance Metrics			Time (s)	
		Top-1	***Top-5***	***Pr***	***Re***	***F1***	**Extract features and Train**	**Test**
Flavia	S3-DDLA + LR	97.92	99.58	0.98	0.98	0.98	284.69	0.02
	DDLA-3 + LR	**98.71**	**99.79**	**0.98**	**0.98**	**0.98**	**239.27**	**0.02**
Folio	S3-DDLA + LR	95.95	99.42	0.96	0.96	0.96	80.2	0.02
	DDLA-3 + LR	**96.38**	**100**	**0.96**	**0.95**	**0.95**	**60.82**	**0.01**
Swedish leaf	S3-DDLA + LR	99.23	100	0.99	0.99	0.99	157.3	0.02
	DDLA-3 + LR	**99.41**	**100**	**0.99**	**0.99**	**0.99**	**80.62**	**0.02**
Leaf-12	S3-DDLA + LR	98.09	100	0.98	0.98	0.98	706.27	0.05
	DDLA-3 + LR	**99.39**	**100**	**0.99**	**0.99**	**0.99**	**541.45**	**0.04**

The ML classifiers used are Linear Discriminant Analysis (LDA), multinomial Logistic Regression (LR), Naive Bayes (NB), Classification and Regression Tree (CART), K-Nearest Neighbor (K-NN), Bagging Classifier

(BC), Random Forest (RFC), Multi-layer Perceptron (MLP) and Support Vector Machine (SVM). Four datasets (Flavia, Folio, Swedish leaf, and custom-developed Leaf-12) are considered to evaluate the DDLAs. The train/test data split ratio used for the experiments is 70:30.

The best results obtained by using similar DLs and dissimilar DLs are listed in Table 6.11. It is observed that the DDLA-3 (MobileNet CNN + DenseNet-121 CNN + LR classifier) resulted in higher accuracy compared to other combinations of architectures for all the datasets. In similar DLs, the S3-DDLA (DenseNet-121 CNN + DenseNet-121 CNN + LR classifier) performed better compared to other combinations, irrespective of the datasets. In addition to Top-1 accuracy, other performance metrics are also listed in Table 6.11. The other performance metrics are Top-5 accuracy, Precision (Pr), Recall (Re), F1-Score (F1), and computation time (Training time and Testing time). The DDLA-3 with LR classifier is exposed to real-time leaf images. It is observed that the number of mispredictions is greatly reduced when compared to single Deep Learning Architectures.

From the experimental analysis, it is identified that the Dual Deep Learning Architecture (DDLA) are highly suitable for real-time plant species recognition. In particular, DDLA-3 (MobileNet CNN + DenseNet-121 CNN) with Logistic Regression (LR) classifier resulted in higher accuracy with lower computation time. The results are consistent for all the considered datasets (Flavia, Folio, Swedish leaf, and Leaf-12).

CHAPTER 7

CONCLUSION AND FUTUREWORK

7.1 CONCLUSION

Real-time plant species recognition is challenging due to the vast diversity of plant species, the arrangement of leaves in plants (simple leaf and compound leaf), variation in geometry, color changes (due to aging or seasonal changes) and image acquisition (Scale, rotation, viewpoints, backgrounds, illumination). Most of the available leaf datasets reported in literatures are related to plant species from different parts of the world (the USA, European Countries, China, Japan, Malaysia, Brazil, Vietnam, etc.). The number of leaf datasets related to Indian plant species is limited. The images in the available dataset are heavily preprocessed. Also, it does not take into account the few scenarios, such as illumination variation, rotational changes, camera viewpoints, scale variations, and different backgrounds. Hence, it is not suitable for real-time recognition of plant species.

In this research work, four different approaches are carried out to perform real-time plant species recognition. The approaches are Conventional method, Neural Network, Single Deep Learning architectures, and Dual Deep Learning Architectures (DDLA). Four datasets are used to evaluate the performance of the plant species recognition system. They are Flavia, Folio, Swedish leaf, and custom-developed Leaf-12 datasets. The Leaf-12 dataset contains more number of images compared to standard datasets (Flavia, Folio, and Swedish leaf). Also, the images with variations (illumination, scale, rotation, camera viewpoints, and backgrounds) are included in the custom-developed dataset. Hence, it is highly appropriate for use in real-time.

7.1.1 Conventional Method for Plant Species Recognition

- A feature extraction method termed as Color, Shape, and Texture Fusion (CSTF) is proposed for Plant Species Recognition (PSR).
- CSTF consists of Color features (Color Channel Statistics), Shape features (Hu Moments (HM)), and Texture features (Local Binary Pattern (LBP) and Haralick Texture (HT)).
- Further, the Machine Learning classifiers (Linear Discriminant Analysis (LDA), multinomial Logistic Regression (LR), Naïve Bayes (NB), K-Nearest Neighbor (K-NN), Classification and Regression Tree (CART), Random Forest Classifier (RFC), Bagging Classifier (BC), and Support Vector Classifier (SVM)) are used for the classification of plant species.
- For Flavia dataset, CSTF+LDA produced highest top-1 and top-5 accuracies of 89.17% and 98.96%. Also, performance metrics such as Precision (*Pr*), Recall (*Re*) and F1-score (*F1*) are 0.91, 0.89 and 0.89.
- CSTF+LDA produced higher accuracies of 79.77% (*top-1*) and 95.38% (*top-5*) for Folio dataset. *Pr*, *Re* and *F1* metrics are 0.84, 0.80 and 0.79, respectively.
- The performance metrics for Swedish leaf dataset using (HT+HM)+LDA are 92.01% (*top-1*), 99.7% (*top-5*), 0.94 (*Pr*), 0.93 (*Re*), and 0.93 (*F1*).
- Leaf-12 dataset showed improved *top-1* and *top-5* accuracies of 82.38% and 99.13% for CSTF+RFC. The performance metrics Pr, Re, and F1 are 0.83, 0.83, and 0.83, respectively.
- For three datasets, the CSTF feature extraction method outperformed other feature combinations.

7.1.2 PReLU (Parametric Rectified Linear Unit) based Backpropagation Neural Network (P-BPNN) for Plant Species Recognition

- Plant species recognition is carried out using the proposed PReLU (Parametric Rectified Linear Unit) activation function based Backpropagation Neural Network (P-BPNN).
- P-BPNN contains an input layer, two hidden layers (500 and 250 neurons), and one output layer.
- Normalized pixel intensities act as input to the P-BPNN.
- Optimal hyperparameters such as Optimizer, Learning rate, Alpha in PReLU, and number of epochs are identified for P-BPNN.
- At 200 epochs, the *Top-1* accuracy, *top-5* accuracy, *Pr*, *Re* and *F1* for Flavia dataset are 85.83%, 98.75%, 0.86, 0.86, and 0.85.
- Folio dataset achieves a *top-1* accuracy of 76.88% at 150 epochs. Other performance metrics such as *top-5* accuracy, *Pr*, *Re* and *F1* obtained are 95.95%, 0.85, 0.77 and 0.78.
- At 200 epochs, the *top-1*, *top-5* accuracies, *Pr*, *Re*, and *F1* for Swedish leaf dataset are 94.97%, 100%, 0.95, 0.95, and 0.95, respectively.
- The *top-1* accuracy for Leaf-12 dataset is 87.5%. *Top-5* accuracy, *Pr*, *Re* and *F1* are 99.41%, 0.88, 0.87 and 0.87 respectively.
- Irrespective of the four datasets considered, Adam optimizer with a learning rate of η=0.001, PReLU activation function with alpha, α=0.1 resulted in higher accuracies. While the number of epochs varied for each dataset.

7.1.3 Single Deep Learning Architectures for Plant Species Recognition

- Plant species recognition is carried out using two approaches of single Deep Learning (DL) architectures (Approach-1: CNN as feature extractor + ML classifier; Approach-2: fine-tuned CNN architecture).
- The utilized pre-trained CNNs are VGG-16, VGG-19, Inception-V3, Inception ResNet-V2, Xception, MobileNet, DenseNet-121, DenseNet-169, and DenseNet-201 architectures.
- Approach-1 uses pre-trained CNNs for feature extractor. The extracted features are classified using ML (LDA, LR, NB, CART, K-NN, RFC, BC, SVM, Multi-Layer Perceptron (MLP)) classifiers.
- Approach-2 trains the dataset using fine-tuned DL architectures.
- Approach-2 showed highest accuracies of 98.58% and 99% for fine-tuned DenseNet-169 CNN on Flavia and Leaf-12 datasets, respectively. Top-5 accuracy, Pr, Re and F1 for Flavia dataset are 99.79%, 0.99, 0.98, and 0.98, respectively. *Top-5* accuracy, *Pr*, *Re* and *F1* for Leaf-12 dataset are 100%, 0.99, 0.99, and 0.99, respectively.
- For Folio and Swedish leaf datasets, both approaches achieved better accuracies.
- DenseNet-121+LR, DenseNet-169+MLP, fine-tuned DenseNet-169 and fine-tuned DenseNet-201 achieved top-1 accuracy of 95.95% on Folio dataset. DenseNet-121+LR required less computational time and so it is chosen as the best for Folio dataset. Other metrics, such as *Top-5*, *Pr*, *Re*, and *F1*, have values of 99.79%, 0.99, 0.98, and 0.98, respectively, when using DenseNet-121+LR.
- Three models namely, MobileNet+LR, fine-tuned DenseNet-121 and fine-tuned DenseNet-169 produced similar *top-1* accuracy of 99.41% for Swedish leaf dataset. MobileNet+LR is chosen as the best model based on the shortest computational time. Other performance metrics, such as

top-5 accuracy, *Pr*, *Re* and *F1* have values of 100%, 0.99, 0.99 and 0.99, respectively.

7.1.4 Dual Deep Learning Architectures (DDLAs) for Plant Species Recognition

- Dual Deep Learning Architectures (DDLAs) are used to improve the performance of real-time plant species recognition.
- Two approaches are followed in DDLAs. They are Similar DDLA and Dissimilar DDLA architectures.
- Similar DDLAs are S1-DDLA (MobileNet CNN+ MobileNet CNN), S2-DDLA (Xception CNN + Xception CNN) and S3-DDLA (DenseNet-121 CNN + DenseNet-121 CNN).
- Dissimilar DDLAs are DDLA-1: MobileNet CNN +Xception CNN; DDLA-2: Xception CNN + DenseNet-121 CNN and DDLA-3: MobileNet CNN +DenseNet-121 CNN).
- DDLAs are used for feature extraction. For classification, the machine learning classifiers (LDA, LR, NB, CART, K-NN, RFC, BC, SVM, and MLP) are applied.
- The performance of DDLA-3 outperformed other similar and dissimilar DDLAs irrespective of the datasets.
- For Flavia dataset, DDLA-3+LR produced *top-1* accuracy of 98.71%. Other metrics, such as *Top-5* accuracy, *Pr*, *Re*, *F1* are 99.79%, 0.98, 0.98 and 0.98, respectively.
- DDLA-3+LR achieved *Top-1* accuracy, *Top-5* accuracy, *Pr*, *Re* and *F1* of 96.38%, 100%, 0.96, 0.95 and 0.95, respectively.
- For Swedish leaf dataset, *top-1* and *top-5* accuracies of 99.41% and 100% were obtained using DDLA-3+LR. Other metrics such as *Pr*, *Re* and *F1* are 0.99, 0.99 and 0.99, respectively.
- *Top-1* accuracy, *top-5* accuracy, *Pr*, *Re* and *F1* obtained for real-time dataset Leaf-12 are 99.39%, 100%, 0.99, 0.99, and 0.99 respectively.

- Irrespective of the four datasets, the DDLA-3 (MobileNet and DenseNet-121) architecture outperformed compared to CSTF, P-BPNN, and single DL methods.
- The number of mispredictions is greatly reduced by using the DDLA-3 + LR classifier, for real-time plant species recognition.

7.1.5 Thesis Summary

Table 7.1 Results obtained for Plant Species Recognition using various approaches (CSTF, P-BPNN, DL, and DDLA) for four leaf datasets

Dataset	Method	Accuracy (%)		Precision, Pr	Recall, Re	F1- Score, F1
		Top-1	Top-5			
Flavia	CSTF+LDA	89.17	98.96	0.91	0.89	0.89
	P-BPNN	85.83	98.54	0.86	0.86	0.85
	Fine-tuned DenseNet-169 CNN	98.58	99.79	0.99	0.98	0.98
	DDLA-3+LR	**98.71**	**99.79**	**0.98**	**0.98**	**0.98**
Folio	CSTF+LDA	79.77	95.38	0.84	0.80	0.79
	P-BPNN	76.88	95.95	0.85	0.77	0.78
	DenseNet-121 + LR	95.95	99.42	0.96	0.96	0.96
	DenseNet-169 + MLP	95.95	99.42	0.97	0.96	0.96
	Fine-tuned DenseNet-169 CNN	95.95	98.27	0.97	0.96	0.96
	Fine-tuned DenseNet-201 CNN	95.95	97.69	0.97	0.96	0.96
	DDLA-3+LR	**96.38**	**100**	**0.96**	**0.95**	**0.95**
Swedish leaf	(HT+HM)+LDA	92.01	99.7	0.94	0.93	0.93
	P-BPNN	94.97	100	0.95	0.95	0.95
	Fine-tuned DenseNet-121 CNN	99.41	100	0.99	0.99	0.99
	Fine-tuned DenseNet-169 CNN	99.41	100	0.99	0.99	0.99
	MobileNet + LR	99.41	100	0.99	0.99	0.99
	DDLA-3+LR	**99.41**	**100**	**0.99**	**0.99**	**0.99**
Leaf-12	CSTF+RFC	82.38	99.13	0.83	0.83	0.83
	P-BPNN	87.50	99.13	0.88	0.88	0.87
	Fine-tuned DenseNet-169 CNN	99.00	100	0.99	0.99	0.99
	DDLA-3+LR	**99.39**	**100**	**0.99**	**0.99**	**0.99**

The summary of performance results obtained by four approaches (Conventional Method, Neural Network, single Deep Learning architecture, and Dual Deep Learning Architecture) are listed in Table 7.1. The performance metrics are obtained by using four datasets (Flavia, Folio, Swedish leaf, and custom-developed Leaf-12).

7.2 FUTURE WORK

Further improvement in plant species recognition system is possible by enhancing the dataset, methodologies, and applications.

7.2.1 Dataset

- Augmented dataset might improve the results of CNNs.
- The Leaf-12 dataset does not contain leaves with white background.
- The leaf images of the Leaf-12 dataset can be increased in size with respect to distant viewpoints. A more robust system could be achieved by increasing the number of rotation of leaf images.
- The self-collected Indian dataset can be increased in terms of the number of medicinal plant species.

7.2.2 Methodologies

- Object Detection algorithms can be used for plant species recognition. It may help in precise, accurate leaf identification and localization.
- Object detection methods can also help in multiple leaf identification.
- Lightweight CNN architectures can be developed for portable applications.

7.2.3 Possible Applications

- CNNs can be employed for accurate prediction of plant disease.
- A mobile application can be developed for potential identification of medicinal plant species.

REFERENCES

1. Aakif, A & Khan, MF 2015, 'Automatic classification of plants based on their leaves', Biosystems Engineering, vol. 139, pp. 66-75.
2. Adinugroho, S & Sari, YA 2018, 'Leaves classification using neural network based on ensemble features', Proceedings of 5th International Conference on Electrical and Electronic Engineering (ICEEE), pp. 350-354.
3. Anderson, LC 1999, 'Collecting and preparing plant specimens and producing an herbarium', Jurnal Department of Biological Science, vol. 20, pp. 295-300.
4. Araújo, VM, Britto, AS, Brun, AL, Koerich, AL & Oliveira, LE 2018, 'Fine-grained hierarchical classification of plant leaf images using fusion of deep models', Proceedings of 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 1-5.
5. Arisdason, W & Lakshminarasimhan, P 2016, 'Status of plant diversity in India: an overview', Central National Herbarium, Botanical Survey of India, Howrah.
6. Atabay, HA 2016, 'A convolutional neural network with a new architecture applied on leaf classification', IIOAB J, vol. 7, no. 5, pp. 226-331.
7. Baldi, P 2012, 'Autoencoders, unsupervised learning, and deep architectures', Proceedings of ICML Workshop on unsupervised and transfer learning, pp. 37-49.
8. Barré, P, Stöver, BC, Müller, KF & Steinhage, V 2017, 'LeafNet: A computer vision system for automatic plant species identification', Ecological Informatics, vol. 40, pp. 50-56.
9. Barré, P, Stöver, BC, Müller, KF, & Steinhage, V 2017, 'LeafNet: A computer vision system for automatic plant species identification', Ecological Informatics, vol. 40, pp. 50-56.
10. Bodhwani, V, Acharjya, DP & Bodhwani, U 2019, 'Deep Residual Networks for Plant Identification', Procedia Computer Science, vol. 152, pp. 186-194.
11. Böhning, D 1992, 'Multinomial logistic regression algorithm', Annals of the Institute of Statistical Mathematics, vol. 44, no. 1, pp. 197-200.
12. Breiman, L 2001, 'Random forests', Machine learning, vol. 45, no. 1, pp. 5-32.

13. Brownlee, J 2017, ‘Gentle introduction to the Adam optimization algorithm for deep learning’, Machine Learning Mastery.

14. Calculate output size of Convolution 2021. Available from: <https://iq.opengenus.org/output-size-of-convolution/>. [29 April 2021].

15. Carranza-Rojas, J & Mata-Montero, E 2016, ‘Combining leaf shape and texture for Costa Rican plant species identification’, CLEI Electronic Journal, vol. 19, no. 1, pp. 1-29.

16. Carranza-Rojas, J, Goeau, H, Bonnet, P, Mata-Montero, E & Joly, A 2017, ‘Going deeper in the automated identification of Herbarium specimens’, BMC Evolutionary Biology, vol. 17, no. 1, p. 181.

17. Carranza-Rojas, J, Mata-Montero, E & Goeau, H 2018, ‘Hidden biases in automated image-based plant identification’, Proceedings of IEEE International Work Conference on Bioinspired Intelligence (IWOBI), pp. 1-9.

18. Cerutti, G, Tougne, L, Mille, J, Vacavant, A & Coquin, D 2013, ‘Understanding leaves in natural images–a model-based approach for tree species identification’, Computer Vision and Image Understanding, vol. 117, no. 10, pp. 1482-1501.

19. Chaki, J & Parekh, R 2011, ‘Plant leaf recognition using shape based features and neural network classifiers’, International Journal of Advanced Computer Science and Applications, vol. 2, no. 10, pp. 41-47.

20. Chaki, J, Parekh, R & Bhattacharya, S 2015, ‘Plant leaf recognition using texture and shape features with neural classifiers’, Pattern Recognition Letters, vol. 58, pp. 61-68.

21. Chollet, F 2017, ‘Xception: Deep learning with depthwise separable convolutions’, Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1251-1258.

22. Chollet, F 2017, Deep learning with Python, Manning Publications, US.

23. Clark, JY, Corney, DP & Tang, HL 2012, ‘Automated plant identification using artificial neural networks’, Proceedings of IEEE Symposium on computational intelligence in bioinformatics and computational biology (CIBCB), pp. 343-348.

24. Classification System 2018, Science Learning Hub. Available from: <https://www.sciencelearn.org.nz/resources/1438-classification-system>. [3 September 2018].

25. Clevert, DA, Unterthiner, T & Hochreiter, S 2015, ‘Fast and accurate deep network learning by exponential linear units (elus)’, Proceedings of the International Conference on Learning Representations, arXiv preprint arXiv:1511.07289, pp. 1-14.

26. Codizar, AL & Solano, G 2016, 'Plant leaf recognition by venation and shape using artificial neural networks', Proceedings of 7th International Conference on Information, Intelligence, Systems & Applications (IISA), pp. 1-4.

27. Common leaf morphologies 2019, Encyclopedia Britannica. Available from:<https://www.britannica.com/science/simple-leaf/media/545336 /374>. [14 Nov 2019].

28. Cope, JS, Corney, D, Clark, JY, Remagnino, P & Wilkin, P 2012, 'Plant species identification using digital morphometrics: A review', Expert Systems with Applications, vol. 39, no. 8, pp. 7562-7573.

29. Dede, MA, Aptoula, E & Genc, Y 2018, 'Deep network ensembles for aerial scene classification', IEEE Geoscience and Remote Sensing Letters, vol. 16, no. 5, pp. 732-735.

30. Deokar, SR, Zope, PH & Suralkar, SR 2013, 'Leaf recognition using feature point extraction and artificial neural network', International Journal of Engineering, vol. 2, no. 1, pp. 1-9.

31. Dias, ROQ & Borges, DL 2016, 'Recognizing plant species in the wild: Deep learning results and a new database', Proceedings of IEEE International Symposium on Multimedia (ISM), pp. 197-202.

32. Du, C & Gao, S 2017, 'Image segmentation-based multi-focus image fusion through multi-scale convolutional neural network', IEEE Access, vol. 5, no. 99, pp. 15750–15761.

33. Du, JX, Huang, DS, Wang, XF & Gu, X 2006, 'Computer-aided plant species identification (CAPSI) based on leaf shape matching technique', Transactions of the Institute of Measurement and Control, vol. 28, no. 3, pp. 275-285.

34. Du, JX, Wang, XF & Zhang, GJ 2007, 'Leaf shape based plant species recognition', Applied mathematics and computation, vol. 185, no. 2, pp. 883-893.

35. Du, JX, Zhai, CM & Wang, QP 2013, 'Recognition of plant leaf image based on fractal dimension features', Neurocomputing, vol. 116, pp. 150-156.

36. Dumoulin, V & Visin, F 2016, 'A guide to convolution arithmetic for deep learning', arXiv preprint arXiv:1603.07285.

37. Eight Inspirational Applications of Deep Learning 2016, Machine Learning Mastery. Available from: <https://machinelearningmastery. com /inspirational-applications-deep-learning/>. [14 July 2016].

38. Floral Statistics of India 2018, ENVIS Centre on Floral Diversity. Available from: <http://bsienvis.nic.in/Database/Floral_Statistics_of _India_2018_26352. aspx>. [24 June 2019].

39. Fu, H & Chi, Z 2006, 'Combined thresholding and neural network approach for vein pattern extraction from leaf images', IEE Proceedings-Vision, Image and Signal Processing, vol. 153, no. 6, pp. 881-892.

40. Gaber, T, Tharwat, A, Snasel, V & Hassanien, AE 2015, 'Plant identification: Two dimensional-based vs. one dimensional-based feature extraction methods', In 10th international conference on soft computing models in industrial and environmental applications, pp. 375-385.

41. Galar, M, Fernández, A, Barrenechea, E, Bustince, H & Herrera, F 2013, 'Dynamic classifier selection for one-vs-one strategy: avoiding non-competent classifiers', Pattern Recognition, vol. 46, no. 12, pp. 3412-3424.

42. Gao, M, Lin, L & Sinnott, RO 2017, 'A mobile application for plant recognition through deep learning', Proceedings of the 13th International Conference on e-Science (e-Science), pp. 29-38.

43. Gaston, KJ & O'Neill, MA 2004, 'Automated species identification: why not?', Philosophical Transactions of the Royal Society of London, Series B: Biological Sciences, vol. 359, no. 1444, pp. 655-667.

44. Ghasab, MAJ, Khamis, S, Mohammad, F & Fariman, HJ 2015, 'Feature decision-making ant colony optimization system for an automated recognition of plant species', Expert Systems with Applications, vol. 42, no. 5, pp. 2361-2370.

45. Ghazi, MM, Yanikoglu, B & Aptoula, E 2017, 'Plant identification using deep neural networks via optimization of transfer learning parameters', Neurocomputing, vol. 235, pp. 228-235.

46. Gogul, I & Kumar, VS 2017, 'Flower species recognition system using convolution neural networks and transfer learning', Proceedings of Fourth International Conference on Signal Processing, Communication and Networking (ICSCN), pp. 1-6.

47. Goodfellow, I, Bengio, Y, Courville, A, Bengio, Y 2016, Deep learning, MIT Press, Cambridge.

48. Goodfellow, I, Pouget-Abadie, J, Mirza, M, Xu, B, Warde-Farley, D, Ozair, S, Courville, A & Bengio, Y 2014, 'Generative adversarial nets', Advances in neural information processing systems, pp. 2672-2680.

49. Gu, J, Wang, Z, Kuen, J, Ma, L, Shahroudy, A, Shuai, B, Liu, T, Wang, X, Wang, G, Cai, J & Chen, T 2018, 'Recent advances in convolutional neural networks', Pattern Recognition, vol. 77, pp. 354-377.

50. Gu, X, Du, JX & Wang, XF 2005, 'Leaf recognition based on the combination of wavelet transform and gaussian interpolation', Proceedings of International Conference on Intelligent Computing, pp. 253-262.

51. Haralick, RM & Shanmugam, K 1973, 'Textural features for image classification', IEEE Transactions on systems, man, and cybernetics, vol. 6, pp. 610-621.

52. Haykin, S 1999, Neural Network-A Comprehensive Foundation, 2nd edition, Pearson, India.

53. He, G, Xia, Z, Zhang, Q, Zhang, H & Fan, J 2018, April. 'Plant Species Identification by Bi-channel Deep Convolutional Networks', In Journal of Physics: Conference Series, vol. 1004, no. 1, p. 012015, IOP Publishing.

54. He, K, Zhang, X, Ren, S & Sun, J 2015, 'Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification', Proceedings of the IEEE international conference on computer vision, pp. 1026-1034.

55. He, K, Zhang, X, Ren, S & Sun, J 2016, 'Deep residual learning for image recognition', Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778.

56. He, Y, Yu, X, Liu, C, Zhang, J, Hu, K & Zhu, HC 2018, 'A 3D Dual Path U-Net of Cancer Segmentation Based on MRI', Proceedings of the IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), pp. 268-272.

57. Howard, AG, Zhu, M, Chen, B, Kalenichenko, D, Wang, W, Weyand, T, Andreetto, M & Adam, H 2017, 'Mobilenets: Efficient convolutional neural networks for mobile vision applications', arXiv preprint arXiv:1704.04861.

58. Hu, J, Chen, Z, Yang, M, Zhang, R & Cui, Y 2018, 'A multiscale fusion convolutional neural network for plant leaf recognition', IEEE Signal Processing Letters, vol. 25, no. 6, pp. 853-857.

59. Hu, MK 1962, 'Visual pattern recognition by moment invariants', IRE transactions on information theory, vol. 8, no. 2, pp. 179-187.

60. Huang, G, Liu, Z, Van Der Maaten, L & Weinberger, KQ 2017, 'Densely connected convolutional networks', Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700-4708.

61. Huang, W, Liu, X, Luo, M, Zhang, P, Wang, W & Wang, J 2019, 'Video-Based Abnormal Driving Behavior Detection via Deep Learning Fusions', IEEE Access, vol. 7, pp. 64571-64582.

62. Jain, SK 1994, Ethnobotany and research in medicinal plants in India, Ethnobot. Search New Drugs, 185, pp. 153-168.

63. Janani, R & Gopal, A 2013, 'Identification of selected medicinal plants leaf using Image Features and ANN', Proceedings of International Conference on Electronic Systems, pp. 238-242.

64. Joly, A, Goëau, H, Bonnet, P, Bakić, V, Barbe, J, Selmi, S, Yahiaoui, I, Carré, J, Mouysset, E, Molino, JF & Boujemaa, N 2014, 'Interactive plant identification based on social image data', Ecological Informatics, vol. 23, pp. 22-34.

65. Jothi Prabha, A & Bhargavi, R 2019, 'Prediction of Dyslexia from Eye Movements Using Machine Learning', IETE Journal of Research, pp. 1-10.

66. Kader, SA 2014, A Textbook of Medicinal Botany, Shamsudheen publishers, Chennai.

67. Kadir, A, Nugroho, LE, Susanto, A & Santosa, PI 2011, 'A comparative experiment of several shape methods in recognizing plants', arXiv preprint arXiv:1110.1509.

68. Kadir, A, Nugroho, LE, Susanto, A & Santosa, PI 2012, 'Performance improvement of leaf identification system using principal component analysis', International Journal of Advanced Science and Technology, vol. 44, pp. 113-124.

69. Kaiser, L, Gomez, AN & Chollet, F 2017, 'Depthwise separable convolutions for neural machine translation', arXiv preprint arXiv:1706.03059.

70. Kala, CP, Dhyani, PP & Sajwan, BS 2006, 'Developing the medicinal plants sector in northern India: challenges and opportunities', Journal of Ethnobiology and Ethnomedicine, vol. 2, no. 1, pp. 1-15.

71. Kalyoncu, C & Toygar, Ö 2015, 'Geometric leaf classification', Computer Vision and Image Understanding, vol. 133, pp. 102-109.

72. Kan, HX, Jin L & Zhou, FL 2017, 'Classification of medicinal plant leaf image based on multi-feature extraction', Pattern Recognition and Image Analysis, vol. 27, no. 3, pp. 581-587.

73. Karpathy, A 2016, 'Convolutional neural networks for visual recognition', Course notes hosted on GitHub. Available from: <http://cs231n. github.io>.

74. Kaya, A, Keceli, AS, Catal, C, Yalic, HY, Temucin, H & Tekinerdogan, B 2019, 'Analysis of transfer learning for deep neural network based plant classification models', Computers and electronics in agriculture, vol. 158, pp. 20-29.

75. Kebapci, H, Yanikoglu, B & Unal, G 2011, 'Plant image retrieval using color, shape and texture features, The Computer Journal, vol. 54, no. 9, pp. 1475-1490.

76. Kheirkhah, FM & Asghari, H 2019, 'Plant leaf classification using GIST texture features', IET Computer Vision, vol. 13, no. 4, pp. 369-375.

77. Kho, SJ, Manickam, S, Malek, S, Mosleh, M & Dhillon, SK 2017, 'Automated plant identification using artificial neural network and support vector machine', Frontiers in Life Science, vol. 10, no. 1, pp. 98-107.

78. Kingma, DP & Ba, J 2014, 'Adam: A method for stochastic optimization', In Proceedings of the International Conference on Learning Representations, arXiv preprint arXiv:1412.6980, pp. 1-15.

79. Kornblith, S, Shlens, J & Le, QV 2019, 'Do better imagenet models transfer better?', Proceedings of the IEEE Conference on computer vision and pattern recognition, pp. 2661-2671.

80. Krause, J, Baek, K & Lim, L 2019, 'A guided multi-scale categorization of plant species in natural images', Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 2639-2647.

81. Krishnapuram, B, Carin, L, Figueiredo, MA & Hartemink, AJ 2005, 'Sparse multinomial logistic regression: Fast algorithms and generalization bounds', IEEE transactions on pattern analysis and machine intelligence, vol. 27, no. 6, pp. 957-968.

82. Larese, MG, Bayá, AE, Craviotto, RM, Arango, MR, Gallo, C & Granitto, PM 2014, 'Multiscale recognition of legume varieties based on leaf venation images', Expert Systems with Applications, vol. 41, no. 10, pp. 4638-4647.

83. Le, TL, Tran, DT & Hoang, VN 2014, 'Fully automatic leaf-based plant identification, application for Vietnamese medicinal plant search', Proceedings of the fifth symposium on information and communication technology, pp. 146-154.

84. Lee, JW & Yoon, YC 2019, 'Fine-Grained Plant Identification using wide and deep learning model', Proceedings of International Conference on Platform Technology and Service (PlatCon), pp. 1-5.

85. Lee, SH, Chan, CS, Mayo, SJ & Remagnino, P 2017, 'How deep learning extracts and learns leaf features for plant classification', Pattern Recognition, vol. 71, pp. 1-13.

86. Li, P, Gong, X, Hu, X, Shi, L, Xue, X, Guo, J, Xu, P & Gan, D 2019, 'Plant Identification Based on Multi-branch Convolutional Neural Network with Attention', Proceedings of Chinese Conference on Image and Graphics Technologies, pp. 472 - 481.

87. Ling, H & Jacobs, DW 2007, 'Shape classification using the inner-distance', IEEE transactions on pattern analysis and machine intelligence, vol. 29, no. 2, pp. 286-299.

88. Liu, Z, Zhu, L, Zhang, XP, Zhou, X, Shang, L, Huang, ZK & Gan, Y 2015, 'Hybrid Deep Learning for Plant Leaves Classification', In International Conference on Intelligent Computing, pp. 115-123.

89. Maas, AL, Hannun, AY & Ng, AY 2013, 'Rectifier nonlinearities improve neural network acoustic models', Proceedings of the 30th International Conference on Machine Learning, vol. 30, no. 1, p. 3.

90. Maden, K 2004, 'Plant collection and herbarium techniques', Our Nature, vol. 2, no. 1, pp. 53-57.

91. Mahdikhanlou, K & Ebrahimnezhad, H 2014, 'Plant leaf classification using centroid distance and axis of least inertia method', Proceedings of 22nd Iranian conference on electrical engineering (ICEE), pp. 1690-1694.

92. Marsland, S 2014, Machine learning: an algorithmic perspective, Chapman and Hall/CRC, Florida.

93. McCulloch-Pitts Neuron — Mankind's First Mathematical Model of a Biological Neuron 2019, Towards Data Science. Available from: <https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1>. [25 April 2019].

94. Minetto, R, Segundo, MP & Sarkar, S 2019, 'Hydra: An ensemble of convolutional neural networks for geospatial land classification', IEEE Transactions on Geoscience and Remote Sensing, vol. 57, no. 9, pp. 6530-6541.

95. Mitchell, TM 1997, Machine learning, McGraw Hill, New York.

96. Mohanraj, V, Sibi Chakkaravarthy, S, Gogul, I, Sathiesh Kumar, V, Kumar, R & Vaidehi, V 2018, 'Hybrid feature descriptors to detect face spoof attacks', Journal of Intelligent & Fuzzy Systems, vol. 34, no. 3, pp. 1411-1419.

97. Munisami, T, Ramsurn, M, Kishnah, S & Pudaruth, S 2015, 'Plant leaf recognition using shape features and colour histogram with K-nearest neighbour classifiers', Procedia Computer Science, vol. 58, pp. 740-747.

98. Murphy, GE & Romanuk, TN 2014, 'A meta-analysis of declines in local species richness from human disturbances', Ecology and Evolution, vol. 4, no. 1, pp. 91-103.

99. Mzoughi, O, Yahiaoui, I, Boujemaa, N & Zagrouba, E 2016, 'Semantic-based automatic structuring of leaf images for advanced plant species identification', Multimedia Tools and Applications, vol. 75, no. 3, pp. 1615-1646.

100. Naresh, YG & Nagendraswamy, HS 2016, 'Classification of medicinal plants: an approach using modified LBP with symbolic representation', Neurocomputing, vol. 173, pp. 1789-1797.

101. Naresh, YG 2016, Machine Learning Approaches for Classification of Medicinal Plants. Ph.D. thesis, University of Mysore.

102. Nguyen, TTN, Le, TL, Vu, H, Nguyen, HH & Hoang, VS 2017, 'A combination of deep learning and hand-designed feature for plant identification based on leaf and flower images', In Asian Conference on Intelligent Information and Database Systems, pp. 223-233.

103. Novotný, P & Suk, T 2013, 'Leaf recognition of woody species in Central Europe', Biosystems Engineering, vol. 115, no. 4, pp. 444-452.

104. Nwankpa, C, Ijomah, W, Gachagan, A & Marshall, S 2018, 'Activation functions: Comparison of trends in practice and research for deep learning', arXiv preprint arXiv:1811.03378.

105. Ocquaye, ENN, Mao, Q, Song, H, Xu, G & Xue, Y 2019, 'Dual Exclusive Attentive Transfer for Unsupervised Deep Convolutional Domain Adaptation in Speech Emotion Recognition', IEEE Access, vol. 7, pp.93847-93857.

106. Ojala, T, Pietikäinen, M & Mäenpää, T 2000, 'Gray scale and rotation invariant texture classification with local binary patterns', In European Conference on Computer Vision, pp. 404-420.

107. Pacifico, LD, Macario, V & Oliveira, JF 2018, 'Plant Classification Using Artificial Neural Networks', Proceedings of International Joint Conference on Neural Networks (IJCNN), pp. 1-6.

108. Pan, J, Sun, H, Song, Z & Han, J 2019, 'Dual-Resolution Dual-Path Convolutional Neural Networks for Fast Object Detection', Sensors, vol.19, no. 14, pp. 1-16.

109. Pan, SJ & Yang, Q 2009, 'A survey on transfer learning', IEEE Transactions on knowledge and data engineering', vol. 22, no. 10, pp.1345-1359.

110. Pang, PK & Lim, KH 2019, 'Review on Automatic Plant Identification Using Computer Vision Approaches', IOP Conference Series: Materials Science and Engineering, IOP Publishing, vol. 495, no. 1, pp. 1-9.

111. Pawara, P, Okafor, E, Surinta, O, Schomaker, L & Wiering, M 2017, 'Comparing Local Descriptors and Bags of Visual Words to Deep Convolutional Neural Networks for Plant Recognition', In ICPRAM, pp. 479-486.

112. Pawara, P, Okafor, E, Surinta, O, Schomaker, L & Wiering, M 2017, 'Comparing Local Descriptors and Bags of Visual Words to Deep Convolutional Neural Networks for Plant Recognition', Proceedings of ICPRAM, vol. 479, p. 486.

113. Pedregosa, F, Varoquaux, G, Gramfort, A, Michel, V, Thirion, B, Grisel, O, Blondel, M, Prettenhofer, P, Weiss, R, Dubourg, V & Vanderplas, J

2011, ‘Scikit-learn: Machine learning in Python’, Journal of machine learning research, vol. 12, pp. 2825-2830.

114. Pedregosa, F, Varoquaux, G, Gramfort, A, Michel, V, Thirion, B, Grisel, O, Blondel, M, Prettenhofer, P, Weiss, R, Dubourg, V & Vanderplas, J 2011, ‘Scikit-learn: Machine learning in Python’, Journal of Machine Learning Research, vol. 12, pp. 2825-2830.

115. Plant Evolution and taxonomy 2019, Botanical Art & Artists. Available from: <https://www.botanicalartandartists.com/plant-evolution-and-taxonomy.html>. [21 November 2019].

116. Priya CA, Balasaravanan, T & Thanamani, AS 2012, ‘An efficient leaf recognition algorithm for plant classification using support vector machine’, Proceedings of International conference on pattern recognition, informatics and medical engineering (PRIME-2012), pp. 428-432.

117. Pundir, AS & Raman, B 2019, ‘Dual Deep Learning Model for Image Based Smoke Detection’, Fire Technology, vol. 55, no. 6, pp. 2419-2442.

118. Rankothge, WH, Dissanayake, DMSB, Gunathilaka, UVKT, Gunarathna, SACM, Mudalige, CM & Thilakumara, RP 2013, ‘Plant recognition system based on Neural Networks’, Proceedings of International Conference on Advances in Technology and Engineering (ICATE), pp. 1-4.

119. Rizk, S 2019, Plant leaf classification using dual path convolutional neural networks. Ph.D. thesis, Notre Dame University-Louaize.

120. Ruder, S 2016, ‘An overview of gradient descent optimization algorithms’, arXiv preprint arXiv:1609.04747.

121. Rusiecki, A 2019, ‘Trimmed categorical cross-entropy for deep learning with label noise’, Electronics Letters, vol. 55, no. 6, pp. 319-320.

122. Rzanny, M, Mäder, P, Deggelmann, A, Chen, M & Wäldchen, J 2019, ‘Flowers, leaves or both? How to obtain suitable images for automated plant identification’, Plant methods, vol. 15, no. 1, pp. 1-11.

123. Saleem, G, Akhtar, M, Ahmed, N & Qureshi, WS 2019, ‘Automated analysis of visual leaf shape features for plant classification’, Computers and Electronics in Agriculture, vol. 157, pp. 270-280.

124. Satti, V, Satya, A & Sharma, S 2013, ‘An automatic leaf recognition system for plant identification using machine vision technology’, International journal of engineering science and technology, vol. 5, no. 4, p.874.

125. Sharma, S & Gupta, C 2015, 'Recognition of plant species based on leaf images using multilayer feed forward neural network', International Journal of Innovative Research in Advanced Engineering (IJIRAE), vol. 2, no. 6, pp. 104-110.

126. Simonyan, K & Zisserman, A 2014, 'Very deep convolutional networks for large-scale image recognition', arXiv preprint arXiv:1409.1556.

127. Sivanandam, SN & Deepa, SN 2007, Principles of Soft Computing, John Wiley & Sons, New Delhi.

128. Smith, AR 1978, 'Color gamut transform pairs', ACM Siggraph Computer Graphics', vol. 12, no. 3, pp. 12-19.

129. Sobitharaj, KKS 1994, Thala Marangal, Sobitham, Madras.

130. Srinivas, S, Sarvadevabhatla, RK, Mopuri, KR, Prabhu, N, Kruthiventi, SS & Babu, RV 2016, 'A taxonomy of deep convolutional neural nets for computer vision, Frontiers in Robotics and AI, vol. 2, p.36.

131. Structure and function of plant tissue 2019. Available from: <https://gitawati.wordpress.com/science-ii/structure-and-function-of-plant-tissue-2/>. [19 November 2019].

132. Sun, Y, Liu, Y, Wang, G & Zhang, H 2017, 'Deep learning for plant identification in natural environment', Computational intelligence and neuroscience.

133. Szegedy, C, Ioffe, S, Vanhoucke, V & Alemi, AA 2017, 'Inception-v4, inception-resnet and the impact of residual connections on learning', Proceedings of Thirty-First AAAI Conference on Artificial Intelligence, pp. 4278-4284.

134. Szegedy, C, Liu, W, Jia, Y, Sermanet, P, Reed, S, Anguelov, D, Erhan, D, Vanhoucke, V & Rabinovich, A 2015, 'Going deeper with convolutions', Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1-9.

135. Szegedy, C, Vanhoucke, V, Ioffe, S, Shlens, J & Wojna, Z 2016, 'Rethinking the inception architecture for computer vision', Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826.

136. Tan, JW, Chang, SW, Kareem, SBA, Yap, HJ & Yong, KT 2018, 'Deep learning for plant species classification using leaf vein morphometric', IEEE/ACM transactions on computational biology and bioinformatics.

137. Tang, Z, Li, X, Song, J, Wei, M & Zhang, X 2017, 'Colour space selection in image hashing: An Experimental Study', IETE Technical Review, vol. 34, no. 4, pp. 440-447.

138. The artificial neural networks handbook: Part 4 2018. Medium. Available from: <https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e>.[11 November 2018].

139. Tieleman, T & Hinton, G 2012, 'Rmsprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural networks for machine learning, vol. 4, no. 2, pp. 26-31.

140. Top 15 Applications that will rule the world in 2018 and beyond 2017, Medium. Available from: <https://medium.com/breathe-publication/top-15-deep-learning-applications-that-will-rule-the-world-in-2018-and-beyond-7c6130c43b01>. [4 October 2017].

141. Valliammal, N 2013, Computer aided plant identification through leaf recognition using enhanced image processing and machine learning algorithms. Ph.D. thesis, Avinashilingam Deemed University for Women.

142. Wable, PB & Chilveri, PG 2016, 'Neural network based leaf recognition', In International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT), pp. 645-648.

143. Wäldchen, J & Mäder, P 2018, 'Plant species identification using computer vision techniques: A systematic literature review', Archives of Computational Methods in Engineering, vol. 25, no. 2, pp. 507-543.

144. Wang, B, Brown, D, Gao, Y & La Salle, J 2015, 'MARCH: Multiscale-arch-height description for mobile retrieval of leaf images', Information Sciences, vol. 302, pp. 132-148.

145. Wang, X, Du, W, Guo, F & Hu, S 2020, 'Leaf Recognition Based on Elliptical Half Gabor and Maximum Gap Local Line Direction Pattern', IEEE Access, vol. 8, pp. 39175-39183.

146. Wang, X, Liang, J & Guo, F 2014, 'Feature extraction algorithm based on dual-scale decomposition and local binary descriptors for plant leaf recognition', Digital Signal Processing, vol. 34, pp. 101-107.

147. Wang, Z, Ma, Y, Cheng, F & Yang, L 2010, 'Review of pulse-coupled neural networks', Image and vision computing, vol. 28, no. 1, pp. 5-13.

148. Wang, Z, Sun, X, Zhang, Y, Ying, Z & Ma, Y 2016, 'Leaf recognition based on PCNN', Neural Computing and Applications, vol. 27, no. 4, pp. 899-908.

149. Weiss, K, Khoshgoftaar, TM & Wang, D 2016, 'A survey of transfer learning', Journal of Big data, vol. 3, no. 1, pp. 1-40.

150. Weldon, TP, Higgins, WE & Dunn, DF 1996, 'Efficient Gabor filter design for texture segmentation Pattern recognition', vol. 29, no. 12, pp. 2005-2015.

151. Yalcin, H & Razavi, S 2016, 'Plant classification using convolutional neural networks', Proceedings of Fifth International Conference on Agro-Geoinformatics (Agro-Geoinformatics), pp. 1-5.

152. Younis, S, Weiland, C, Hoehndorf, R, Dressler, S, Hickler, T, Seeger, B & Schmidt, M 2018, 'Taxon and trait recognition from digitized herbarium specimens using deep convolutional neural networks', Botany Letters, vol. 165, no. 3-4, pp. 377-383.

153. Zeiler, MD 2012, 'ADADELTA: an adaptive learning rate method', arXiv preprint arXiv:1212.5701.

154. Zhang, C, Zhou, P, Li, C & Liu, L 2015, 'A convolutional neural network for leaves recognition using data augmentation', Proceedings of IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, pp. 2143-2150.

155. Zhang, H, Yanne, P & Liang, S 2012, 'Plant species classification using leaf shape and texture', Proceedings of International Conference on Industrial Control and Electronics Engineering, pp. 2025-2028.

156. Zhang, S & Chau, KW 2009, 'Dimension reduction using semi-supervised locally linear embedding for plant leaf classification', Proceedings of International conference on intelligent computing, pp. 948-955.

157. Zhang, S & Lei, YK 2011, 'Modified locally linear discriminant embedding for plant leaf recognition', Neurocomputing, vol. 74, no. 14-15, pp. 2284-2290.

158. Zhang, S & Zhang, C 2017, 'Plant species recognition based on deep convolutional neural networks', Proceedings of International Conference on Intelligent Computing, pp. 282-289.

159. Zhang, S, Wang, H & Huang, W 2017, 'Two-stage plant species recognition by local mean clustering and Weighted sparse representation classification', Cluster Computing, vol. 20, no. 2, pp. 1517-1525.

160. Zhang, S, Zhang, C, Wang, Z & Kong, W 2018, 'Combining sparse representation and singular value decomposition for plant recognition', Applied Soft Computing, vol. 67, pp. 164-171.

161. Zhao, C, Chan, SS, Cham, WK & Chu, LM 2015, 'Plant identification using leaf shapes—A pattern counting approach', Pattern Recognition, vol. 48, no. 10, pp. 3203-3215.

162. Zhao, ZQ, Xie, BJ, Cheung, YM & Wu, X 2014, 'Plant leaf identification via a growing convolution neural network with progressive sample learning', Proceedings of Asian Conference on Computer Vision, Springer, Cham, pp. 348-361.

163. Zhao, ZQ, Xie, BJ, Cheung, YM & Wu, X 2014, 'Plant leaf identification via a growing convolution neural network with progressive sample learning', Proceedings of Asian Conference on Computer Vision, pp. 348-361.

164. Zhu, H, Liu, Q, Qi, Y, Huang, X, Jiang, F & Zhang, S 2018, 'Plant identification based on very deep convolutional neural networks', Multimedia Tools and Applications, vol. 77, no. 22, pp. 29779-29797.

165. Zhu, X, Zhu, M & Ren, H 2018, 'Method of plant leaf recognition based on improved deep convolutional neural network', Cognitive Systems Research, 52, pp. 223-233.

166. Zhu, Y, Sun, W, Cao, X, Wang, C, Wu, D, Yang, Y & Ye, N 2019, 'TA-CNN: Two-way attention models in deep convolutional neural network for plant recognition', Neurocomputing, vol. 365, pp. 191-200.

167. Zulkifli, Z, Saad, P & Mohtar, IA 2011, 'Plant leaf identification using moment invariants & general regression neural network', Proceedings of 11th International conference on hybrid intelligent systems (HIS), pp. 430-435.

LIST OF PUBLICATIONS

International Journals

1. **Anubha Pearline Sundara Sobitha Raj** & Sathiesh Kumar Vajravelu 2019, 'DDLA: Dual Deep Learning Architecture for Classification of Plant Species', IET Image Processing, vol. 13, no. 12, pp. 2176–2182. ISSN: 1751-9659 (Annexure I). (Impact Factor: 1.995)

2. **Anubha Pearline**, **S**, Sathiesh Kumar, V & Harini, S 2019, 'A study on plant recognition using conventional image processing and deep learning approaches', Journal of Intelligent & Fuzzy Systems, vol. 36, no. 3, pp. 1997-2004. ISSN: 1064-1246 (Annexure I). (Impact Factor: 1.815)