



JIMMA UNIVERSITY
JIMMA INSTITUTE OF TECHNOLOGY
FACULTY OF ELECTRICAL AND COMPUTER
ENGINEERING
MASTERS OF SCIENCE IN COMPUTER ENGINEERING

Medicinal Plant Part Identification and Classification
Using Deep Learning

By Misganaw Aguate

**A Thesis Report is Submitted to School of Graduate Studies of Jimma
University in Partial Fulfillment for the Degree of Master of Science in
Computer Engineering**

Jimma, Ethiopia
March, 2021 G.C

JIMMA UNIVERSITY
JIMMA INSTITUTE OF TECHNOLOGY
FACULTY OF ELECTRICAL AND COMPUTER
ENGINEERING
MASTERS OF SCIENCE IN COMPUTER ENGINEERING

Medicinal Plant Part Identification and classification Using
Deep Learning

By Misganaw Aguate

Advisor: Dr. Abebe Tesfahun

Co-advisor: Mr. Fetulhak Abdurahman

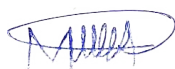
Jimma, Ethiopia

March, 2021 G.C


Declaration

I declared that this research entitled as **Medicinal Plant Parts Identification and Classification using Deep Learning** is my original work and has not been worked, submitted, and presented for any degree of profession in Jimma University or any other universities and institutions. All sources and references used for this study are appropriately acknowledged.

Student Name: **Misganaw Agaute**

Signature:  _____

Main Advisor: **Dr. Abebe Tesfahun**

Signature:  _____

Co-Advisor: **Mr. Fetulhak Abdurahman**

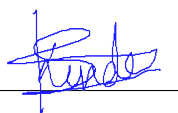
Signature: _____

This Thesis has been Approved by Board of Examiners

External Examiner: **Dr. Dereje Teferi**

Signature:  _____

Internal Examiner: **Dr. Kinde Anley**

Signature:  _____

Chair Person: **Mr. Fetulhak Abdurahman**

Signature: _____

Abstract

In Ethiopia there are a lot of medicinal plants that can cure different types of diseases using their different parts. These plants are often missed by modern medicinal science; they are mainly known by the people who are an experts on indigenous medicine. This study proposes a fine-tuned model to classify the medicinal plant parts. The fine-tuning technique on Mobile Net, VGG16, and InceptionV3 are applied to extract plant features and classify the medicinal part. The batch size, learning rate, and optimizers are tuned to make the models achieve high efficiency during prediction of medicinal plant parts. For classification task, Softmax function is used at the last layer of the CNN. Metrics such as, precision, recall, and F1-Score are used to evaluate the models. A high-resolution camera for data acquisition and google Colab for training and testing are used. When analyzing the experimental result, Mobile Net perform better with an accuracy of 99.84% for training sets and 99.44% for testing sets using learning rate of $1e^{-4}$, optimizer of Adamax, and a batch size of 32. VGG16 performs 99.78% for training sets and 99.37% for testing sets using a learning rate of $1e^{-4}$, Adamax and batch size of 128. InceptionV3 performs 96.12% for training sets and 90.53% for testing sets. While evaluating models using F1_score metric, Mobile Net obtain appreciated performance by scoring 99.44% using optimizer of Adamx and batch size of 32. Without batch normalization at fully connected layer, Mobile Net scores 99.27% using Adamax. Generally Mobile Net gain the best performance using a learning rate of $1e^{-4}$, epoch of 30, batch size of 32, and optimizer Adamax. In this study, Mobile Net is confirmed as the fastest model to train, obtained higher performance, and is suitable to classify the medicinal plant part. This is not due to the small number of convolutional layer rather Mobile Net use depthwise separable convolutional layer to decrease computational complexity (reduce the depth of output feature map by decreasing scalar multiplication through convolution).

Acknowledgment

In the first place, my gratitude is for Dr. Abebe Tesfahun for his motivation and willingness to lead me in this study and giving me supportive idea. Secondly, I greatly acknowledge Mr. Fetulhak Abdurahman and Dr. Getachew Alemu who raised initiative support and recognition in this study area. I would like to say thanks for Dr. Kinde Anley, Dr, Taye Tolu, and Eng. Kris Calpotura who raised a very good suggestion on this study. My gratitude also goes to Mr. Girma Mamo head of the Pharmacy department and Mr. Tamrat Tiqas researcher on Medicinal plants in the Pharmacy department at Jimma University Medical College who informed me about the current situation of medicinal plants in their faculty by correlating his idea with my research questions. The other great gratitude is for Mr. Aregu Yitayew, Uztaz Nuru Jemal, Mr. Yigzaw Beqele, Mr. Aguate Widneh, Merigeta Dele Yman, Merigeta Getachew Mihret, and Merigeta Bekalu (Menkr) Tamiru who have 5 up to 30 years of experiences on herbal medicine preparation. Researchers in phatobiological study of Addis Ababa University also deserve a great thanks. I would like to say thanks to my class mates (friends) who gave me their passionate comments.

Contents

1	Introduction	1
1.1	Background of the Study	1
1.2	Statement of the Problem	3
1.3	Objectives	4
1.3.1	General objective	4
1.3.2	Specific objectives	4
1.4	Research methodology	4
1.5	Significances of the Study	6
1.6	Scope of the Study	7
1.7	Application Area of the Study	7
1.8	Organization of the Paper	8
2	Literature Review	9
2.1	Preliminary Note	9
2.2	Medicinal Plants	9
2.3	Approaches for Medicinal Plant Identification	11
2.3.1	Machine Learning	11
2.3.2	Deep Learning	11
2.3.3	Convolutional Neural Network	13
2.3.4	Transfer Learning	13
2.4	Activation Functions	14
2.4.1	Rectified Linear Unit	14
2.4.2	Softmax	15

2.5	Optimizers	16
2.5.1	Adam	16
2.5.2	Adagrad	16
2.5.3	RMSprop	16
2.6	Learning Rate	17
2.7	Batch Normalization	17
2.8	Batch Size	17
2.9	Dropout	17
2.10	Related Works	18
2.10.1	Machine Learning	18
2.10.2	Deep Learning	19
2.11	Research Gaps and Solutions	23
3	System Design and Methodology	24
3.1	Preliminary Note	24
3.2	Proposed Method	24
3.3	Data Acquisition	26
3.4	Image Annotation	27
3.5	Implementation of Image Annotation in the System	28
3.6	Pre-processing	29
3.7	Feature Extraction	29
3.7.1	How CNN Extract Features	30
3.8	Proposed CNN Model	30
3.8.1	Feature Learning	31
3.8.2	Classification	33
3.9	Loss Function	33
3.10	Multi Class Setup	33
3.11	Transfer Learning	34
3.11.1	Mobile Net	35
3.11.2	VGG16	36
3.12	Inception_V3	36

3.13	Model Evaluation Metrics	36
3.13.1	Precision	36
3.13.2	Recall	37
3.13.3	F1_score	37
3.13.4	Micro-Average	37
3.13.5	Macro-Average	38
3.13.6	Weighted Average	38
3.14	Prediction	38
3.15	Requirements	39
4	Experimental Results and Discussions	40
4.1	Preliminary Note	40
4.2	Dataset Variation Visualizations Using Histograms	40
4.2.1	Light Intensity Variation	40
4.3	Training and Testing Dataets	42
4.4	Hyper-parameter Tuning	42
4.5	Training and Testing Models	43
4.5.1	Training and Testing Accuracy	43
4.5.2	Mobile Net	43
4.5.3	VGG16	44
4.5.4	Inception_V3	45
4.6	Comparisons Among Fine-tuned Models	46
4.7	Classification Report	47
4.8	Model Evaluations	49
4.8.1	F1 Measurement	49
4.9	Experiment without Batch Normalization	50
4.10	Computational Time Variation Among Models	52
4.11	Predictions	53
5	Conclusion and Recommendations	55
5.1	Conclusion	55
5.2	Recommendations	57

A	Fine-tuned Mobile Net	63
A.1	Learning Rate 0.0001	63
B	VGG16	65
B.1	Learning Rate 0.0001	65
B.2	Inception_V3	66
B.3	Learning Rate 0.0001	66
C	Training and Testing Accuracy/loss Comparison Among Models	68
C.1	Experiemnt One	68
C.2	Experiemnt Two	69
D	Model Evaluation	72
D.1	F1-Score, Precision and Recall using Learning Rate 0.0001	73
D.2	F1-Score, Precision and Recall using Learning Rate 0.001	74
D.3	Accuracy and Loss for Training and Testing Datasets	75
E	Sample Fragmented Python Code	81
E.1	Pre-processing	81
E.2	Load image features and splitting target labels	81
E.3	Classification Reports	82
E.4	Micro, Macro, and Weighted Average	82

List of Figures

2.1	Structure of deep neural network	12
2.2	Structure of Model development in deep learning	12
2.3	Architecture of convolutional neural network	13
2.4	Rectified linear unit [?]	15
2.5	Research gap in deep learning	23
3.1	System procedure	24
3.2	Overall system Architecture	25
3.3	Dataset distribution among target classes	26
3.4	Sample leaf images	28
3.5	Difference between RGB and gray scale image	29
3.6	Feature extraction in the internal layers of CNN	30
3.7	CNN architecture of the proposed model	31
3.8	Convolution process of CNN	32
3.9	Max polling	32
3.10	Operational components of the Model	33
3.11	Alternatives in transfer learning technique	35
4.1	Intensity variation in the dataset	41
4.2	Intensity level of sample 1	41
4.3	Intensity level of sample 3	41
4.4	Accuracy and loss of Mobile Net for training and testing dataset in the best case	44
4.5	Testing accuracy of the models using BS = 32 and LR = $1e^{-4}$	46
4.6	Testing Loss of the models using BS = 32 and LR = $1e^{-4}$	46

4.7 Confusion matrix obtained from Mobile Net 47

4.8 Overall F1_score for tested models 50

4.9 Testing Accuracy of the models without BN 51

4.10 Testing loss of the models without BN 51

4.11 Prediction results of sample medicinal plant using Mobile Net 54

A.1 Position index for screen shoot images 63

A.2 Mobile Net accuracy/loss graph using batch size 32 and learning rate $1e^{-4}$. . . 63

A.3 Mobile Net accuracy/Loss using batch size 64 and learning rate $1e^{-4}$ 64

A.4 Mobile Net accuracy/loss using batch size 128 and learning rate $1e^{-4}$ 64

B.1 VGG16 accuracy/loss using batch size 32 and learning rate $1e^{-4}$ 65

B.2 VGG16 accuracy/loss using batch size 64 and learning rate $1e^{-4}$ 65

B.3 VGG16 accuracy/loss using batch size 128 and learning rate $1e^{-4}$ 66

B.4 Inception_V3 accuracy/loss using batch size 32 and learning rate $1e^{-4}$ 66

B.5 Inception_V3 accuracy/loss using batch size 64 and learning rate $1e^{-4}$ 67

B.6 Inception_V3 accuracy/loss using batch size 128 and learning rate $1e^{-4}$ 67

C.1 Training and testing accuracy comparison using BS 64 and LR $1e^{-4}$ 68

C.2 Training and testing loss comparison using BS 64 and LR $1e^{-4}$ 68

C.3 Training and testing accuracy comparison using BS 128 and LR $1e^{-4}$ 69

C.4 Training and testing loss comparison using BS 128 and LR $1e^{-4}$ 69

C.5 Training and testing accuracy comparison using BS 32 and LR $1e^{-3}$ 69

C.6 Training and testing loss comparison using BS 32 and LR $1e^{-3}$ 70

C.7 Training and testing accuracy comparison using BS 64 and LR $1e^{-3}$ 70

C.8 Training and testing loss comparison using BS 64 and LR $1e^{-3}$ 70

C.9 Training and testing accuracy comparison using BS 128 and LR $1e^{-3}$ 70

C.10 Training and testing loss comparison using BS 128 and LR $1e^{-3}$ 71

D.1 F1 measure using BS 64, 128 and LR $1e^{-4}$ 72

E.1 Pre-processing 81

E.2 Load image features and splitting target labels 81

E.3 Classification report 82

E.4 Evaluation metrics 82

List of Tables

2.1	Summary of Literatures	22
3.1	Samples of target label mapping	28
3.2	Requirements	39
4.1	Hyperparameter tuning values	42
4.2	Mobile Net training and testing accuracy using LR $1e^{-4}$ with different BS . . .	43
4.3	Mobile Net training and testing accuracy using LR $1e^{-3}$ with different BS . . .	44
4.4	VGG16 training and testing accuracy using LR $1e^{-4}$ with different BS	45
4.5	VGG16 training and testing accuracy using LR $1e^{-3}$ with different BS	45
4.6	InceptionV3 training and testing accuracy using LR $1e^{-4}$ with different BS . . .	45
4.7	Analysis of correctly classified and misclassified classes from the confusion matrix	48
4.8	Performance of Mobile Net on individual classes	49
4.9	Performance difference between the absence and the presence of batch normalization in FC layer of the models	51
4.10	Model and hyperparameters selection summary	52
4.11	Computational time variance among models	53
D.1	Mobile net F1-Score, Precision and Recall using LR $1e^{-4}$ with different BS . . .	73
D.2	VGG16 F1-Score, Precision and Recall using LR $1e^{-4}$ with different BS	73
D.3	Inception_V3 F1-Score, Precision and Recall using LR $1e^{-4}$ with different BS .	74
D.4	Mobile Net F1-Score, Precision and Recall using LR $1e^{-3}$ with different BS . .	74
D.5	VGG16 F1-Score, Precision and Recall using LR $1e^{-3}$ with different BS	75
D.6	Inception_V3 F1-Score, Precision and Recall using LR $1e^{-3}$ with different BS .	75

D.7 Mobile Net training and testing accuracy using LR $1e^{-4}$ with different BS . . . 76

D.8 VGG16 training and testing accuracy using LR $1e^{-4}$ with different BS 76

D.9 Inception_V3 training and testing accuracy using LR $1e^{-4}$ with different BS . . . 77

D.10 Mobile Net training and testing accuracy using LR $1e^{-3}$ with different BS . . . 77

D.11 VGG16 training and testing accuracy using LR $1e^{-3}$ with different BS 78

D.12 Inception_V3 training and testing accuracy using LR $1e^{-3}$ with different BS . . . 78

D.13 Selected medicinal plants 79

D.14 Selected medicinal plants continued 80

Acronyms

AF	Activation Function
BS	Batch Size
DL	Deep Learning
DT	Decision Tree
FC	Fully Connected
GLCM	Gray Level Correlation Matric
HOG	Histogram Oriented Gradient
HS	Horizontal shift
KNN	K-nearest neighbor
LBP	Local Binary Pattern
LR	Learning Rate
PCA	Principal Component Analysis
RELU	Rectified Linear Unit
RGB	Reed, Green and Blue
SVM	Support Vector Machine
VS	Vertical Shift
ZM	Zernike Moment

Chapter 1

Introduction

1.1 Background of the Study

Plants have been used as direct medicinal sources since ancient times. Even today, plant based pharmacopeias continue to play an essential role in world health care. The WHO also defined herbal medicine as finished labeled medicinal products that contain as active ingredients in aerial or underground parts of plants or other plant parts or combinations [1]. Research indicates that, Ethiopia is rich with more than 6,500 species of vascular plants out of which an estimated 12% are endemic and about 887 species are used as medicinal plants. More than 95% of traditional medicinal preparations are made from plant origin [1].

Plants are used as antimalarial, insecticidal, and repellents in the Ethiopian traditional herbal medicine system [41]. About 44% of herbal medicine preparations were reported to be obtained from roots [3]. Most of the remedies are prepared from a single species and are mainly taken orally [41]. Root was the most frequently used part in remedy preparation [5]. Plants are identified based on leaves, flowers, bark, seed, fruits, roots, stem, and others [7].

The physiological parts of plants like leaves, flowers, fruits, seeds, barks, and roots are used as key ingredients and compositions of herbal medicines. Hence, the digitization of information about useful plant species is necessary. Several researchers have tried to develop a robust and efficient plant recognition system by exploiting pattern recognition and image processing techniques based on plant leaves, flowers, barks, and fruits. However, leaves play a very important role compared to the other parts of a plant as it contains rich information as well as more

medicinal value [8]. Other parts of the plant needs a complex task to extract unique features because if root is taken, it needs taking the image from multi direction when compare to leaf image. So, more unique features can be obtained from leaves without complexity compared to other parts of plants. From leaf, a lot of unique features can be extracted for identification and classification of medicinal plant parts such as veins, margin (contour), texture, size, color, shape, topographical structure (including width, height, perimeter, area) and others.

This study used pre-trained CNN architecture to save computational time by achieving good performance with small number of data and training epochs. This architecture is used to extract plant features and train the models. Manually extracting single leaf feature and maximum of two or three features vector as usual machine learning technique causes to the model operates in a low efficiency. Because if the leaves are affected by environmental and atmospheric factors, it may miss useful information and create ambiguity for the trained model. Hence, a combination of different leaf features and automatic feature extraction should be used to solve this problems and avoid similarity of plants or herbs in the same species. Deep Learning is also called Deep Structured Learning. Deep Learning is used in Computer Vision and Image Identification, etc. Deep Learning uses Artificial Neurons that imitates human Neurons. Deep Neural Networks (DNN), Deep Belief Networks, and Recurrent Neural Network, these are used in Speech Recognition, Natural Language Processing, Machine Translations, Audio Recognitions, bioinformatics, Drug design, Medical Image Identifications, and Medicinal plant identification and classification [23].

The identification of the medicinal plant part can be done using image processing and chemical ingredient extraction. But using its chemical ingredient is a tedious task because to test a given plant for curing purpose, physicians should go to find experts, need well organized and documented resource, to extract all part of plant ingredients and give the result to the designed model to check either that plant part has the medicinal advantage or not. So, it is time-consuming and needs high-cost expenditure for laboratory equipment. So, to reduce this problem identification of the medicinal plant part using image processing is the preferred approach. That is why in this study image processing is used for the identification and classification of medicinal plant parts.

1.2 Statement of the Problem

In Ethiopia, to manufacture medicinal drugs, it needs improvement of indigenous medicinal knowledge. To achieve this, active ingredient extraction from medicinal plants is necessary which again needs identification of specific parts of medicinal plants. Researchers in the area of pharmacy and plant science suffer from challenges to conduct their researches on medicinal plants due to the absence of the automatic computing system. Data collected from Jimma University through interviews. From this interview department of pharmacy and plant Science indicate that researchers are face challenges to conduct their study on medicinal plants. Some of the challenges are, finding experts, absence of well-organized and standardized documented resources, and close similarity among the same species of plants. The previous and even current trend of researches in those departments consumes a lot of time to identify specific parts of the medicinal plant before ingredient extraction. As the interviewee said that, they refer ordinary known medicinal plants from orally documented sources. So, they need experts to know the real physical structure of medicinal plants. Close similarity of plants in a species creates confusion on the thought of researchers to identify the one that has medicinal use unless they should take all plants in to experiment sequentially. Now a days to automate the medicinal plant identification process, a lot of researches are conducted using machine learning but nothing is done to identify specific parts of the plant. This identification work takes time, financial resource, and needs experts. Even though medicinal plant identification is relevant for ingredient extraction, nothing is done to alleviate the problem. Hence, it is necessary to design medicinal plant parts identification system to reduce the need of experts, more time, and money for the improvement of traditional knowledge to modern medicinal science. In this study the following research question are answered:

1. What could be the possible techniques to improve indigenous knowledge of medicinal plant part identification and classification for modern medical science?
2. What would be the optimal training and model hyper parameters to classify medicinal plant parts?
3. What would be the best deep learning model to achieve desirable performance for identification and classification of medicinal plant parts?

1.3 Objectives

1.3.1 General objective

The objective of this research is to develop a model for medicinal plant part identification and classification using deep learning.

1.3.2 Specific objectives

- Collect medicinal plant leaf image data and label the corresponding parts.
- Implement image size reduction and normalization to create trainable data
- Select appropriate deep learning models and apply deep learning hyper-parameters to get optimal performance of classification model.
- Fine-tune pre-trained models to extract unique features of leaves, train it with the data.
- Test and evaluate performances of the models and select the best model that suits with the study target.

1.4 Research methodology

In this research methodology principles, practices, and scientific procedures are applied to solve a specific problem. This research mainly follows a design science research (DSR) methodology that needs the creation of innovative and purposeful artifacts for a specific problem domain. In the case of this study, the specific problem domain is the identification of medicinal plant parts. This artifact solves the problem that is not yet been solved which is identifying specific parts of the medicinal plants. This research is conducted to gain insight that can be applied to the feature design project. The common process of the methodology used in this study are in the format of Peffers *et al.* [9] such as problem identification, objectives of the solution, design and development, demonstration, evaluation, and communication.

Problem Identification

To define the research problems properly, primary information is collected from the department of pharmacy and researchers in Jimma University through an interview regarding the process of medicinal plant identification. Socioeconomic problems regarding the absence of improvement of indigenous knowledge to modern medicinal drug production are identified. Relevant literature on medicinal plant identification techniques is reviewed. The weaknesses with the current medicinal plant identification method are discussed based on the literature reviewed.

Objectives of the Solution

Based on the problem statement and relevant background, the objectives of the solution are explored with the underlying inquiry of what would be the artifact accomplished. The objectives are drawn out to develop a deep learning model for medicinal plant parts identification and classification to make the two initial procedures of ingredient extraction cost-effective, low effort, and time manageable. These two steps are medicinal plant identification and its healer part specifications. To achieve these identification and classification processes, specific tasks are selected that are used later to evaluate the designed model by comparing the observed results.

Design and Development

This stage focuses on creating of artifact solutions including the methods that rely on the combination of the problem statement, a review of the relevant background, and the objectives of the solution. The design includes the way of data collection, data annotation, data preprocessing, feature extraction and training of models with input data using transfer learning technique. Hence, the deep learning model is developed based on Keras API (Application program interface) using Google Colab by writing python code.

Demonstration

In this phase design science research (DSR) can be described in two ways. The first one is looking at the development of artifacts that part of the research is designed fully practical. Which means demonstrating the research in the real world. The second way and employed by this study are gaining insight that can be applied to the feature design project. Hence, this

research has conducted demonstration that is applied in the real world. The study simply tried to show (give insight) what the medicinal plant parts classification result looks when it will be demonstrated in the real world by predicting samples of medicinal plants.

Evaluation

In this phase how well the method (artifact) supports the solution to the problems are observed and measured by comparing the objectives with actual analyzed results from the use of the methods. The quality and efficiency of the models are rigorously tested and evaluated based on experimental results by quantifying it out of 100 percent with well-executed evaluation methods (dynamic analysis). These methods are accomplished using model evaluation metrics such as precision, recall, and F1_score.

Communication

This is the last phase of this research to present the problems in current researches, societies, and concerned organizations. This is also to present the importance of the solution drawn using the artifact, the artifact's novelty, efficiency, and effectiveness to researchers and other relevant audiences (practicing professionals).

1.5 Significances of the Study

The major task of this study is the classification of the specific parts of the medicinal plant to easily use indigenous knowledge in modern medical science. The following are some of the significances of this study.

- Pharmaceutical and botanical physicians can identify medicinal plants easily without the need of an expert. Due to this Cost and time will reduce to extract ingredients.
- Standardized medicinal plant identification will be available, accordingly ambiguity due to the similarity of medicinal plant species will reduce.
- Used as a basic tool for researchers and medicinal drug preparation. That means before any ingredient extraction it is used as plant parts identifier..

- Plant images are taken from the backside to consolidate vein feature extraction. Because vein has an essential features in the identification of medicinal plants.
- Plant leaf image captured using different focal length depending on the broadness of leaves.
- Multi class classification introduced for the specific parts of the medicinal plants
- Local plant leaf images collected and annotated for the first time in Ethiopia. Hence, it will be available for the future researchers.

1.6 Scope of the Study

This study proposed for identification and classification of the medicinal plant parts by integrating the applications of computer vision and deep learning techniques. The study mainly covered the idea of transfer learning (fine-tuned model) and testing of how to identify and classify the medicinal plant parts. So, the climax goal here is identifying and classifying parts of medicinal plants to make ingredient extraction process easy, cost effective and time manageable and confirm the specified part is really used as a source of medicines. The study does not incorporate the corresponding disease cured by the specified plant parts.

1.7 Application Area of the Study

This study is mainly applicable for pharmaceutical and botanic laboratory. A person who may be a pharmacist or other concerned body may asks about the requirement of an expert to know the specific part of a given plant to extract ingredients. At this time he can use the designed intelligent system. The system takes plant image as input, identifies input image and specify part of the plant. Then, the pharmacist can extract ingredient from identified medicinal plant part to know the percentage of medicinal ingredients present in its parts. Finally, the physician can define that medicinal plant part has active ingredients to cure diseases.

1.8 Organization of the Paper

The thesis is organized in the following manner: In chapter two a review of literature for every method of the thesis work is presented. In the subsequent chapter, a discussion of CNN and related concepts is presented. The methodology, different hyper-parameters, and materials are discussed in this chapter. An in depth result and discussion of the study implemented in this study are described in chapter four. Finally, chapter five concludes and gives recommendations for future work.

Chapter 2

Literature Review

2.1 Preliminary Note

In this section, previously studied researches are discussed regarding to feature extraction and classification methods for the identification and classification of the medicinal plants. Each study has its own drawback and advantages. The previous studies are reviewed by making a category as classical machine learning and deep learning. The study also tried to identify the gaps and in the review the number of datasets, methods, and models that authors used are discussed. This section also reviews about medicinal plants and the approaches by which medicinal plant identifications are manipulated.

2.2 Medicinal Plants

In the world there is around 35,000 up to 70,000 species of plants that are used for medicinal purposes [1]. Medicinal plants also play a crucial role and constitute the backbone of traditional system of medicine practices. Over 100 years ago, the development and massive production of chemically synthesized drugs have revolutionized healthcare in most parts of the world. However, large parts of the society in developing countries still depends on the traditional experts and herbal medicines for their remedial action. In Africa up to 90% and in Ethiopia 80% of the population depend on traditional medicine to help meet their health care needs [1]. Most of the treatments are prepared from a single species and are mainly taken orally [41]. Hence, overdose and underdose may be occur while utilizing traditional medicine. Through technological

research and development, countries of the Far East Asia such as China, South Korea and India have been able to meet 75% of their health care needs through the development and utilization of herbal medicine and traditional medicine practice [1]

In 2002 an ethnobotanical survey was carried out to collect information on the use of medicinal plants by the Zay people who live on islands [41]. The study showed that from the major component of plant parts harvested leaf takes first rank. The majority of the Ethiopian people including the Meinit ethnic group are highly dependent on medicinal plants for their day-to-day public healthcare and veterinary of needs [5]. Hence, it needs to document and analyze the knowledge. From the collected data, root is the most frequently used part of the plants in first aid preparation.

The integration of the traditional knowledge to modern medicinal science is found to be the backbone for the consistency of traditional medical knowledge and practice in the study area. There for, Efforts should be made to incorporate traditional medicine in school and universities curriculum to make younger people appreciate its advantage. Plant species are documented as medicinal along with the traditional knowledge on utilization of plant resources by local people in and around Tara Gedam and Amba Forests in Libo Kemkem Wereda, South Gondar Zone. A total of 175 plant species have been identified. Most of them have medicinal properties in their leaf, bark, root, stem, flower, seed and fruits. The knowledge transfer system is quite restricted within the family. Hence, users were found less aware of conservation of medicinal plants and its advantages. It is essential to apply them to modern knowledge of science and technology to meet the ever-increasing requirements of humankind [10].

In 2013 a total of 135 medicinal plant species belonging to 128 genera and 71 botanical families were reported to treat human diseases in the Ankober, North Shewa Zone. About 44% of preparations were reported to be obtained from roots [3]. In 2015 a rich assemblage of plants used as antimalarials, insecticidal and repellents in the Ethiopian traditional herbal medicine system was compiled. A total of 177 plant species belonging to 148 genera and 65 families were documented. The uses of leaves and roots are reported higher frequencies than other parts [2]. Plant medicines have been commonly used over the years for treatment and prevention of diseases and health promotion. However, a systematic approach does not yet

develop to assess their quality, safety and effectiveness. Hence, research is necessary for filter out active ingredients and develop new medicinal products for the common usage. To achieve these researches, the primary action should be applied on **identification of specific parts of medicinal plant** that have active ingredients in the way of cost effective, small effort, and time manageable which needs conducting another technological research and projects.

2.3 Approaches for Medicinal Plant Identification

2.3.1 Machine Learning

Machine Learning is a modeling technique that involves data. Machine Learning is a technique that figures out the model based on data [11]. The data may be documents, audio, video, and images. The data that used by Machine Learning in the modeling process called as training data. Machine Learning approach cannot achieve the optimal goal with the wrong training data. The same concept applies to Deep Learning. Therefore, it is critical for Machine Learning approaches to obtain unbiased training data that adequately reflects the characteristics of the field data.

Machine Learning explores the study and construction of algorithms that can learn from data and make predictions it. These algorithms operate by building a model from sample inputs in order to make data driven predictions or decisions, rather than following strictly static program instructions [12]. Some of the elements present in machine learning are learning, algorithms, data, and modeling. Its compliment fields are statistics, computational intelligence, data mining, data science and artificial intelligence. In its special sub fields there are a variety of learning techniques such as supervised, unsupervised, semi supervised, reinforcement, and a special type called as deep learning. It is possible to solve classification, clustering, optimization, and regression problems by using machine learning.

2.3.2 Deep Learning

Deep Learning is a Machine Learning technique that adopt the deep neural network. It is possible to say that deep neural network is the multi-layer neural network constructed using two or more hidden layers [11]. Figure 2.1 Shows the basic structure of a multi-layer neural

network that contains three layers such as, the input layer, hidden layers, and the output layer. The input layer receives input raw data to the network. The hidden layer receives data from the input layer and create a unique feature map for individual leaf images. The output layer receives mapped (extracted) feature from the hidden layers and start classification depending on the target labels (true value). At the output layer, y_1 , y_2 and y_3 represent the value of the target label (classes) to be predicted. The deep neural network operates in the place of the final

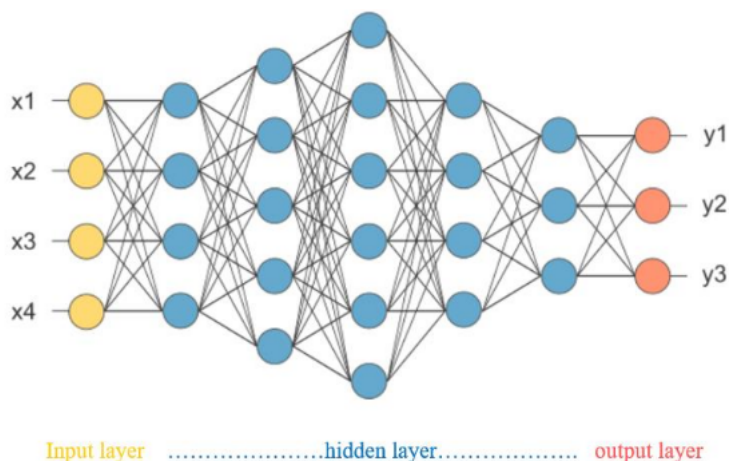


Figure 2.1: Structure of deep neural network

product of Machine Learning. Hence, in figure 2.2 the learning rule becomes the algorithm that produces the model (the deep neural network) based on the training data [11]. Deep Learning means simply the use of a deeper neural network that have a lot of hidden layers.

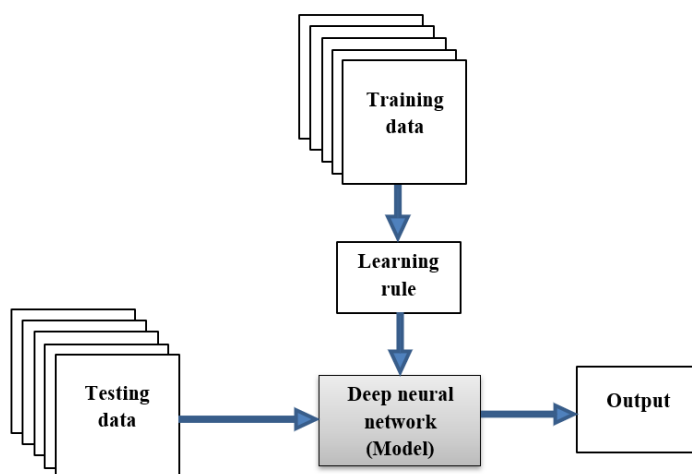


Figure 2.2: Structure of Model development in deep learning

2.3.3 Convolutional Neural Network

CNN is employed by deep learning and it is mainly designed for object detection and image classification. Convolutional Network includes the feature extractor in the training process rather than designing it manually. The feature extractor of Convolutional Network is composed of special kinds of neural networks in which the weights are determined via the training process [11]. The primary feature and advantage of Convolutional Network is improvement of the manual feature extraction design into the automated process. Convolutional Network contains a neural network that extracts features of the input image and another neural network that classifies the feature image. The feature extraction layer has combined pairs of the convolution and pooling layers [11]. Figure 2.3 indicates the basic structure of CNN. The input images

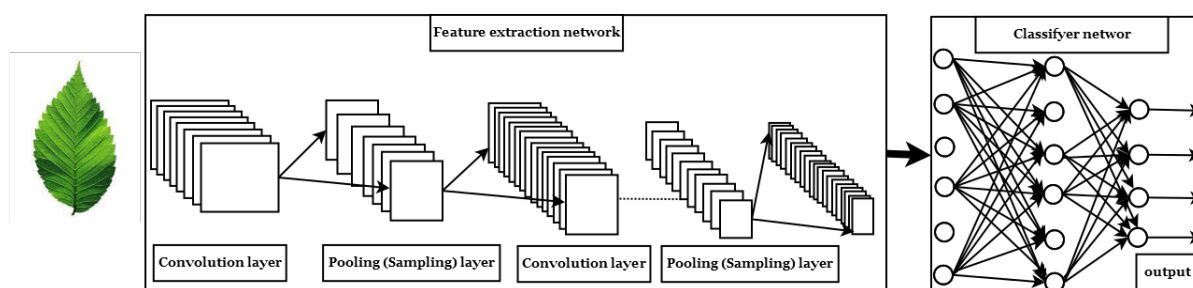


Figure 2.3: Architecture of convolutional neural network

are feed into the feature extraction network. Feature extraction network computes convolution, ReLU, and max pooling activity to create a unique feature map for individual leaf images. The extracted feature map then pass to the classification neural network. The classification neural network then processes based on the features of the image computed by feature extraction and finally produce the output.

2.3.4 Transfer Learning

Transfer learning refers to identifying and applying the knowledge accumulated from previous tasks to new tasks from a related domain. The key feature here is the ability to identify the commonality between the domains [13]. Transfer learning also the area in machine learning that aims to utilize the knowledge gained while solving one problem to solve a different but related problem [13].

2.4 Activation Functions

Activation functions (AF) are used in neural networks to compute the weighted sum of input and biases, which is used to decide if a neuron can be fired or not. It manipulates the presented data through some gradient processing usually gradient descent and afterward produces an output for the neural network, that contains the parameters in the data. These AFs are often referred to as a transfer function in some literature. The output of this AF can be given by:

$$y = (w_1 * x_1 + w_2 * x_2 + \dots + w_n * x_n + b) \quad (2.1)$$

However, since the output is linear in nature, the nonlinear activation functions are required to convert these linear inputs to non-linear outputs [25]. These AFs are transfer functions that are applied to the outputs of the linear models to produce the transformed non-linear outputs, ready for further processing. The non-linear output after the application of the AF is given by [25]:

$$y = \alpha(w_1 * x_1 + w_2 * x_2 + \dots + w_n * x_n + b) \quad (2.2)$$

Where α is the activation function. The position of an AF in a network structure depends on its function in the network thus:

- When the AF is placed after the hidden layers, it converts the learned linear mappings into non-linear forms for propagation.
- While in the output layer, it performs predictions.

Most of the time ReLU and Tanh are used in a base convolutional layer of the neural network and SoftMax and Sigmoid is utilized at the last layer of the network to make classification decision. In Softmax, each target label probability to be a winner is dependent on remaining probability values [25].

2.4.1 Rectified Linear Unit

The rectified linear unit (ReLU) activation function was proposed by Nair and Hinton 2010, and ever since, has been the most widely used activation function for deep learning applications with state-of-the-art results to date [?]. The ReLU is a faster learning AF, which has

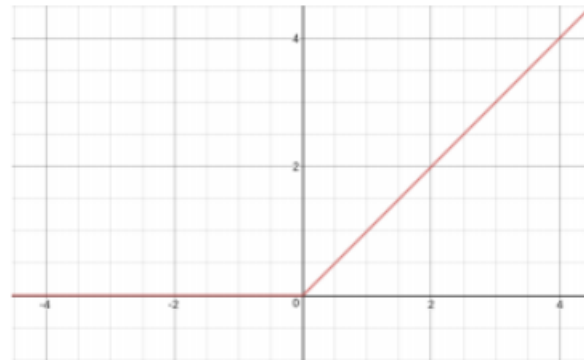


Figure 2.4: Rectified linear unit [?]

proved to be the most successful and widely used function [?]. The ReLU represents a nearly linear function and therefore preserves the properties of linear models that made them easy to optimize, with gradient-descent methods [?]. The ReLU activation function performs a threshold operation to each input element where values less than zero are set to zero thus the ReLU is given by [?]:

$$f(x) = \max(0, x) = \begin{cases} x_i, & \text{if } x_i \geq 0 \\ 0, & \text{if } x_i < 0 \end{cases} \quad (2.3)$$

2.4.2 Softmax

Softmax is used in neural computing to compute probability distribution from a vector of real numbers and produces an output which is a range of values between 0 and 1. Since multi-class models are adopted, this function is used where it returns probabilities of each class, with the target class having the highest probability [25].

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^k e^{x_j}} \quad (2.4)$$

Where 'f' is softmax, 'xi' is input vector, 'e^{xi}' is standard exponential function for input vector, 'e^{xj}' is standard exponential function for output vector, 'k' number of classes in multi class classification.

2.5 Optimizers

Optimizers are the algorithm that are applied in deep learning neural network to adjust the weight parameters of each neuron to get small values of loss while performing training of the model. Some of these algorithms include Adam, Adamax, Adagrad, Adadelata, RMSprop, SGD [27] and etc.

2.5.1 Adam

This Optimizer is designed to combine the advantages of two recently popular methods called AdaGrad which works well with sparse gradients, and RMSProp (Tieleman and Hinton, 2012), which works well in on-line and non-stationary settings. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods. This optimizer is computationally efficient, has little memory requirements, is invariant to the diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters [28].

2.5.2 Adagrad

AdaGrad is the most straightforward improvement to SGD. AdaGrad adjusts the learning rate dynamically based on the historical gradient in some previous iterations. One main benefit of AdaGrad is that it eliminates the need to tune the learning rate manually when compare to SGD [29].

2.5.3 RMSprop

In RMSprop, the idea is to consider not accumulating all historical gradients, but focusing only on the gradients in a window over a period, and using the exponential moving average to calculate the second-order cumulative momentum. This optimizer is developed to resolve the radically diminishing learning rates of AdaGrad [29].

2.6 Learning Rate

The learning rate is an important parameter in the training process [30]. It controls the rate (speed) during learning process of the models. Specifically, it controls the amount of proportioned error then the weights of the model are updated with each time as they are updated. This is happen at the end of each batch of training samples. If the learning rate is high, the modifications in the weights are bigger that is bigger steps are taken and it make the loss on training dataset oscillate over training epochs. When the learning rate is very small, training speed is slower and may permanently stop with a high training error ??.

2.7 Batch Normalization

Batch normalization is proposed to reduce internal covariate shift, and in doing so dramatically accelerates the training of deep neural network. It accomplishes this via a normalization step that fixes the means and variances of layer inputs. Furthermore, batch normalization regularizes the model and reduces the need for Dropout. Batch Normalization achieves the same accuracy with 14 times fewer training steps and beats the original model by a significant margin [31].

2.8 Batch Size

The batch size means the number of training examples used in one iteration. For each batch, the gradients will be computed and updates will be made to the weights of the network accordingly. Normally the training continues until validation performance deteriorates [32].

2.9 Dropout

Dropout is one type of convolutional neural network layer in deep learning to overcome the problem of overfitting while training a deep neural network. It is implemented by making some layer of a neural network inactive by dynamically removing certain connections existing between the nodes randomly [33]. Dropout means that some units (hidden and visible) in the neural network are dropped out temporarily, that is, are removed from the network [34]. The

dropped-out units do not take part in the forward pass and backpropagation [35].

2.10 Related Works

2.10.1 Machine Learning

In 2019 the extraction of shape, color and texture features from leaf images was employed and train an artificial neural network (ANN) to identify the exact leaf classes [16]. The researchers are tried to extract leaf features from color, shape and texture and train it using ANN in both separate features and combination of features. The neural network returned accuracy results of 55%, 72% and 92% using shape, color and texture respectively. The combination of shape, color and texture for 63 leaf images resulted in a correct leaf identification rate of 94.4% with a minimum of eight input features. However, vein and contour play a relevant role for robust plant part classification and the technique of deep learning rather than classical machine training to make it automatic and compute for a large number of data sets [16]. A system also proposed in [17] which can identify the plant species based on the input leaf sample. An improved vegetation index, ExG-ExR is used to obtain more vegetative information from the images. Therefore, the ExG-ExR index identifies a binary plant region of interest. The authors are used color and texture features to classify medicinal plant using Logistic Regression. The researchers used 100 data set that is 20 images per class. The fixed zero threshold, ExG-ExR vegetative index is successfully tested for image data set. Finally they got classifier accuracy of 93.3%. But the study used very small data set. Even though the classifier is classical it should have optimum number of data set. Only color and texture feature can not give consistent classification because these feature may suffer from environmental and atmospheric changes. For relevant classification in addition to color and texture other leaf features with combination is recommended.

A new medicinal plant data set developed based on the extraction of texture and color features from plant leaf images [18]. A complete automatic plant recognition system is proposed. For the recognition step, five well established classifiers from machine learning and plant classification literature are employed: Decision Tree classifier, K-Nearest Neighbors classifier,

Weighted K-Nearest Neighbors classifier, Random Forest classifier and a Multi-Layer Perceptron trained with Back-propagation algorithm. To evaluate the selected classifiers, an empirical analysis complemented by a hypothesis test of type Friedman/Nemenyi test in relation to four classification metrics (testing accuracy, testing precision, testing recall and testing f-measure) is adopted. MLP-BP and RFC are able to achieve the best performances concerning all selected classification metrics according to Friedman/Nemenyi hypothesis test, in comparison to the other selected classifiers.

In [19], a methodology presented to identify herbal species using their flower images. Researcher extract color, shape and different texture features from flower images and created three different feature vectors with Haralick, Tamura and Gabor textures. Then classify each with different classifiers such as SVM, Decision Trees and K-NN. For the shape features, has only considered roundness and aspect ratio from flower images. The authors conclude that SVM can be considered as a well-performed classifier with herbal flowers and Gabor texture can be considered as a good texture feature rather than Tamura and haralick textures. Texture has a major impact to identify flower species and the over all accuracy of the designed classifier is 86.52%.

All the above proposed classical machine learning classifier uses leaf features of shape, color, texture either separately or by combination to classify medicinal plant but these features are not a distinct attribute of leaves rather vein [21] and contour feature can preferable to classify medicinal plants with acceptable performances.

2.10.2 Deep Learning

In 2017 a knowledge transfer from object identification to plant species identification was proposed where the raw plant leaf image is represented into deep features. The authors presented an efficient technique for leaf acquisition. The image is transformed to device independent color space that is further used to compute VGG-16 feature map. This feature map is re-projected to PCA subspace to optimize the performance for species recognition. To prove the robustness, the study uses two different types of plant leaf data sets such as ICL and medicinal data set. The author used VGG-16 ConvNet architecture to train and classify with combina-

tion of PCA which can reduce feature vector to optimize classification cost. Finally they got classification accuracy of 94.1% using VGG-16 and PCA which is the proposed classifier technique [20].

The researchers in [21] were extracted useful leaf features directly from the raw representations of input data using Convolutional Neural Networks (CNN), and gain intuition of the chosen features based on a Deconvolutional Network (DN) approach. They show that CNN trained on whole leaves and leaf patches exhibit different contextual information of leaf features. They categorize them into global features that describe the whole leaf structure and local features that focus on venation. A new leaf data set, named as the MalayaKew (MK) Leaf Data set 1 consisting of 44 classes is employed in the experiment. Using the whole leaf image, the researcher found the emergence of global features describing the holistic structure of leaf such as the shape, color, texture and margin. While using leaf patches they noticed that CNN tends to capture the local intrinsic patterns of venation. Importantly, these findings demonstrate that CNN trained with leaf patches is capable of recognizing the relevant vein patterns and differentiating them among species without needing any manual segmentation or pre-processing on veins. The author reused AlexNet pre-trained model. Finally they get an accuracy of 99.95% for local features and 97.7% for global features when applying on MLP classifier. But the authors stated that they can not conclude leaf vein can classify medicinal plant efficiently because this study takes vein feature from patches of leaf part. This leads to miss classification because testing patches may different from training patches while implementing it for real world utilization.

In [22] a new CNN-based method named D-Leaf proposed. The leaf images were pre-processed and the features were extracted by using three different Convolutional Neural Network (CNN) models namely pre-trained AlexNet, fine-tuned AlexNet and D-Leaf. These features were then classified by using five machine learning techniques, namely, Support Vector Machine (SVM), Artificial Neural Network (ANN), k-Nearest Neighbour (KNN), Naïve-Bayes (NB) and CNN. The D-Leaf model achieved a comparable testing accuracy of 94.88% as compared to AlexNet (93.26%) and fine-tuned AlexNet (95.54%) models. In addition, CNN models performed better than the traditional morphometric measurements (66.55%). Perfor-

mance of the designed method was validated on three publicly available data sets, which are, MalayaKew, Flavia and Swedish Leaf Data set. Besides that, the researchers in this study found CNN is a favorable feature extraction method rather than a classification method. As the author investigated ANN models achieved the best performance so ANN is highly suitable with the CNN feature extraction model. But classical ANN have lack of number of data set and gradient descent vanishing problem to perform accurate classification using computing machine.

AyurLeaf which is a Deep Learning based Convolutional Neural Network (CNN) model proposed in [23], to classify medicinal plants using leaf features such as shape, size, color, texture etc. The proposed data set contains leaf samples from 40 medicinal plants. A deep neural network inspired by AlexNet is utilized for the efficient feature extraction from the data set. Finally, the classification is performed using CNN and SVM classifiers. The designed model achieved a classification accuracy of 96.76% on AyurLeaf data set. A model(Deep Neural Networks) for the identification of medicinal plants was built [24]. To train the model researchers are used around 8,000 images belonging to four different classes. They had made their training by considering the images taken under different lighting conditions. This property is adopted in order to produce a very good classification accuracy when tested in real time scenario. In this research the whole work is implemented from scratch and produces an accuracy percentage of 85.15%.

Table 2.1 describes about the precise overview of previously studied researches. The table indicates the number of data sets, features extracted from plant leaves, techniques used to extract features and train the models, and finally the performance of the models out of 100 percent.

Table 2.1: Summary of Literatures

Year	Title	Datasets	Features	Techniques	Performances
2013	Identification of selected medicinal plant using image feature and ANN	63	Shape, color and texture	GLCM and ANN	94.4
2019	Real time identification of medicinal plant using machine learning	1000	Color and texture	LR, EXG and EXR	93.3
2019	Automatic classification of medicinal plant species based on color and texture feature	1148	Color and texture	DT, KNN, WKNN, MLP	97.5
2019	Texture dominant approach for identifying Ayurveda herbal species using flower	1700	Shape, color and texture	SVM, DT and KNN	86.52
2017	Medicinal plant information using deep features	1700	Shape and texture	CNN with PCA	VGG16 94.1
2018	Deep learning for plant species classification using vein morphometric	1990	39 vein features	SVM, ANN, KNN, NB and CNN	D-leaf 94.88, Alex Net 93.3, Fine-tuned Alex Net 94.54
2019	Deep learning approach for classification of medicinal plant	1200	Shape, size, color and texture	CNN and SVM	CNN 95.06, SVM 96.76
2019	Identification of medicinal plant and their usage using deep learning	8000	All features	CNN	85.15

2.11 Research Gaps and Solutions

In this section, previously studied researches are discussed regarding to feature extraction and classification methods for the identification and classification of the medicinal plants. To remind some of these gapes, all studies that are done previously cannot answer the question which specific part of the medicinal plants are used to cure human diseases? Rather all study focuses on simple identification of plant that recommends which plant has medicinal use. The reviewed study also misses the curing potential of medicinal plants using more than one part. This study answered this question by mapping specific parts of the plant on corresponding leaf image during annotation phase. Figure 2.5 describes how a physician utilize an automatic system for

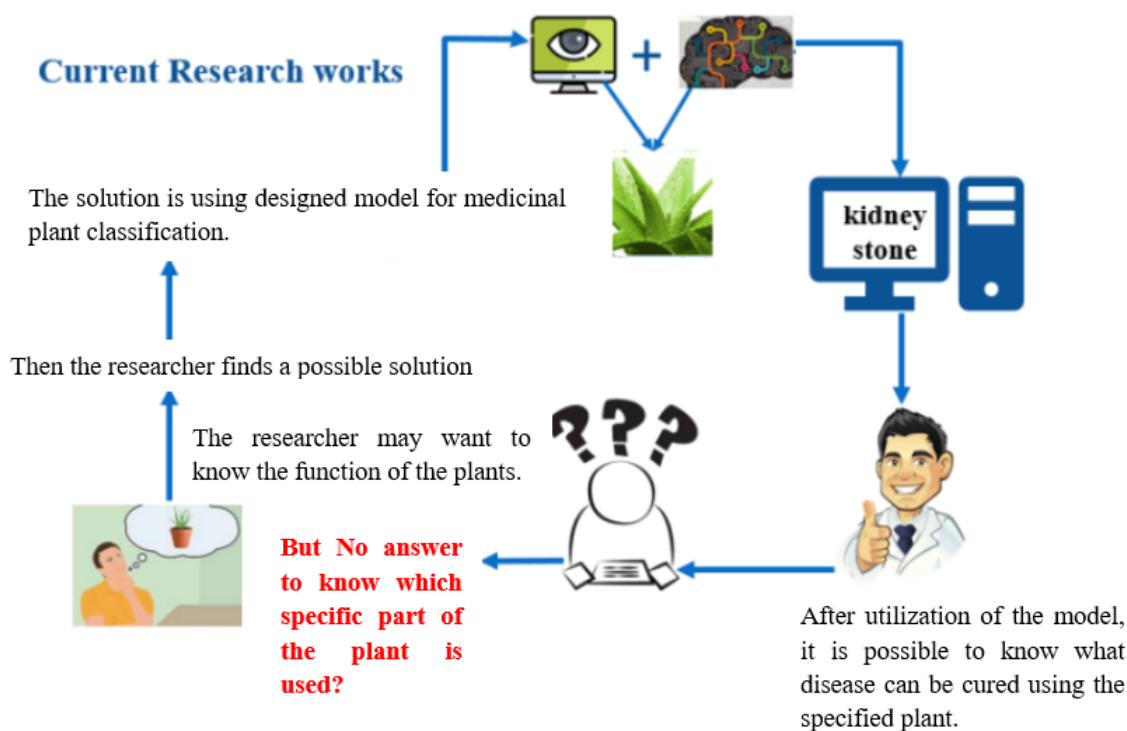


Figure 2.5: Research gap in deep learning

identification of medicinal plant. A person who may be a pharmacist or other concerned body may want to know the function of a given plants towards medicinal purpose. When a person tried to find a possible solution, he can use the designed intelligent system. The system takes plant image as input, identifies and specify which disease is cured by the plant. But the system does not give any clue about the specific part of the plant. Hence, again he ask which specific part is used? and does get nothing answer for this question.

Chapter 3

System Design and Methodology

3.1 Preliminary Note

This section discuss about the methods used to achieve the objectives of the study. It discuss about how data are collected and annotated, how image preprocessing is handled, data is split, features are extracted, and models are trained and evaluated. This section also tried to elaborate about the unique feature of the study to classify specific medicinal plat parts using multi class technique and how to apply it on fine-tuned models with locally collected data.

3.2 Proposed Method

The proposed method has incorporated the following Seven major tasks: Data collection, Annotation, Preprocessing, Feature extraction, Training, Evaluation, and Prediction. The high-level view of the overall system is shown in the figure 3.1. First, medicinal plant leaves captured

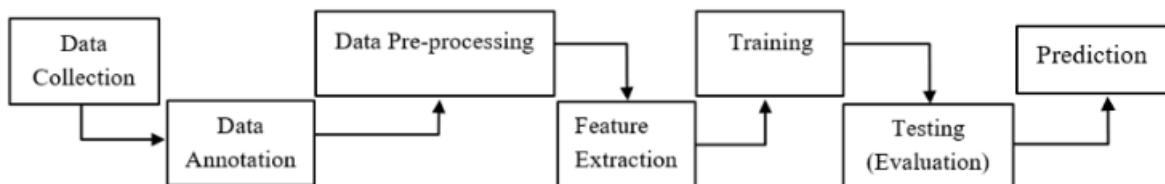


Figure 3.1: System procedure

and other parts are mapped to leaves image as a target classes. Second, the preprocessing of images. Third, trainable data created by changing numerical pixel value into matrix using a spe-

cial python function called a NumPy array. Fourth, features are extracted using the fine-tuned model. This feature extraction process is the internal processes done by CNN. Fifth, identification and classification of the Medicinal plant is done by using Softmax classifier. Sixth, model performance is tested and the best which fit with the proposed solution is selected, and Seventh prediction with new plant experimented. Figure 3.2 contains two main parts such as image data

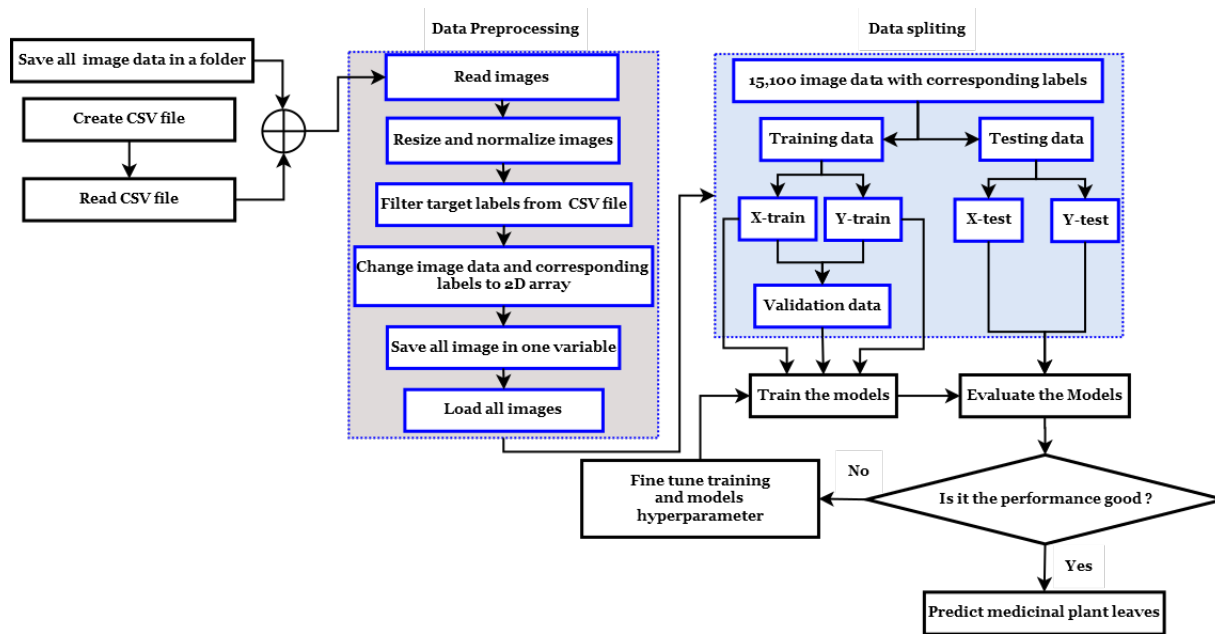


Figure 3.2: Overall system Architecture

processing and target label processing. On the image processing side, after putting image data in a folder and reading the CSV file, these two files are concatenated to read a full image file. From a folder, the original image of leaves taken and from the CSV file, the corresponding filename of the image taken is labeled in the first column of the CSV file. The input image is resized to 128X128 and the pixel values are normalized between 0 and 1 to reduce computation time during training. The image pixels are changed into a series of a 2D array to be understandable by computer machines (models) while processing it. The array values of image pixel with the corresponding mapped (labeled) classes are saved as one variable and later reload to save computational time when these data are needed later. Two of the columns are dropped from the CSV file which holds image name and label tags, and leave the remaining column as the target label of corresponding image files. Figure 3.2 indicates the image data and corresponding target labels is split into training and testing set as X-train, X-test, and Y-train, Y-test to evaluate the models. Here 20% of the image data and target label are taken for the test set for one time

cross validation purpose. Training set is used to train the model and the testing set is unseen data during training but used later to test the model. Further 20% from X-train and Y-train data are taken the validation set to cross check whether the model train well or not within each epoch during training. The next task is fitting X-train, Y-train, and validation data to the model and train it. Here fine tuned Mobile Net, VGG16, and Inception-V3 are used on the train model section. Then we test the efficiency of the models using X-test Y-test. During this procedure the validity of the model are evaluated and then if the models are performed good, predictions of medicinal plant parts are takes place otherwise tuning of training and model hyperparameters are applied and train the model again until good result is obtained.

3.3 Data Acquisition

The data is collected using a high-resolution camera (TECHNO SPARK K7 with 13MP and SAMSUNG A30S with 25MP). These sample data are taken from Gojjam, Jimma, and Kefa. These places are selected due to the presence of variety of plant species and selected plant is easily accessible. The researches done at Addis Ababa University are used as initial source of data. These data collection procedures took two months. One week, two weeks, and one month in Kefa, Jimma, and Gojjam respectively. When taking images of the plant leaves, to avoid data augmentation, the different light intensity strength and direction of leaf positions are considered. Figure 3.3 shows how many sample data are present among the corresponding

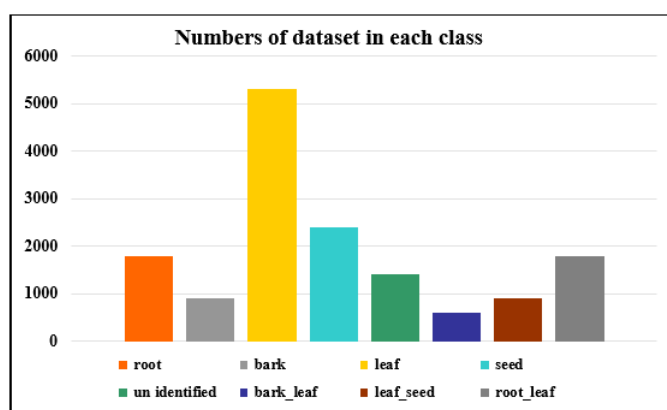


Figure 3.3: Dataset distribution among target classes

categories. This figure shows the graph of the distribution of the dataset in each class. This

implies that the dataset in this study has unbalanced data among each of the class (category). The numbers of the data are categorized in leaf, root, seed, root-leaf, bark, leaf-seed, bark-leaf, and unidentified with the amount of 5,300, 1800, 2400, 1800, 900, 900, 600, and 1400 respectively. This shows the dataset contains 15,100 medicinal plant part images. Each plant has an equal number of leaf images (300 images) but the number of plants categorized in each class is different. That is why an unbalanced amount of data among the categories (classes) occurred. While performing the data acquisition activity, the following requirement is drawn out as guidance for data acquisition:

- Medium aged and young parts of the plant leaf are selected
- The focal length of the camera (distance between leaves and camera) is taken depending on the broadness and narrowness of the plant leaves.
- The white background of the image was selected to avoid confusion and get a clear leaf image structure.
- To reduce loss of vein feature, dried and diseased leaves excluded to make the prediction consistent.
- For the same purpose like the above rule, the leaf images are captured immediately after cut out from the plants to reduce the loss of leaf features.
- The camera stands perpendicular to the leaf stand. i.e. the position taken 90° from the leaf stand to avoid shadow of veins on leaf surfaces.

3.4 Image Annotation

After collecting the medicinal plant image, the next procedure is annotation (labeling) of each plant part. These labeling tasks are supported by five experts with 5, 10, 13, 28- and 30-year of experience who are currently working on traditional medicine preparation in to conducting research. The researchers have tried to label the medicinal plant parts in their pathobiological study at Addis Ababa University. The procedure is, first easily available plants are selected from research papers and books with the corresponding labels. To make sure about the data

labeling, selected plants are brought to the experts. Then plant parts are labeled with the help of those five experts. Finally, the plant parts that are commonly labeled by three experts are taken as the complete data set to train the models. Initially, 72 plants are selected. But finally, 52 are taken as the final data set. 20 plants are removed because there are no commonly labeled parts by the experts for those plants.

3.5 Implementation of Image Annotation in the System

This study takes plant leaves as identity of medicinal plant parts such as, root, leaf, bark, and seed. The scenario is mapping (labeling) each plant part on the corresponding leaf images. The main thing is knowing the image and specific parts of the plant. Figure 3.4 and table 3.1 describe more about the implementation of this annotation for the designed model.

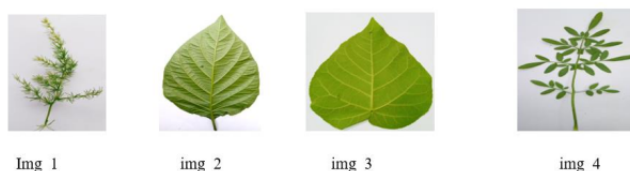


Figure 3.4: Sample leaf images

Table 3.1: Samples of target label mapping

File name	Tags	root	bark	leaf	Seed	Un identified	bark_leaf	leaf_seed	root_leaf
img_1	['root']	1	0	0	0	0	0	0	0
img_2	['leaf_seed']	0	0	0	0	0	0	1	0
img_3	['bark_leaf']	0	0	0	0	0	1	0	0
img_4	['seed']	0	0	0	1	0	0	0	0

Let us take img_1, it is a leaf image to be fed into the designed model. This image is labeled as shown in the table 3.1, then when the system gets leaf image the same as the previously trained image, in this case, img_1, the model classifies it as root. Here image file is stored in a folder with the name of img_1 and on the annotation part called imge_1 and is labeled as shown on the above CSV file. Then the image file and CSV file are concatenated at the time of preprocessing. So, the model uses a leaf image to classify img_1 as root because root is labeled (mapped) to

this image feature. The opportunity here is assuring this annotation is based on research and the expert currently working on the preparation of traditional medicine from medicinal plant parts.

3.6 Pre-processing

Pre-processing plays a crucial roles to extract features from medicinal plant leaves and train the model. It starts from resizing a variety of input images and making each suitable for the designed model. The size of all input images are resized to 128x128 in 2D to reduce computation time. Even though the color feature consumes computational time in CNN to process the image, RGB image is used here to reduce loss of useful color features. In figure 3.5 shown on the left, has three channels which takes more time to compute while the gray image, shown on the right, has only one channel which takes less computational time, i.e. colored image takes 3 times more processing power and time.

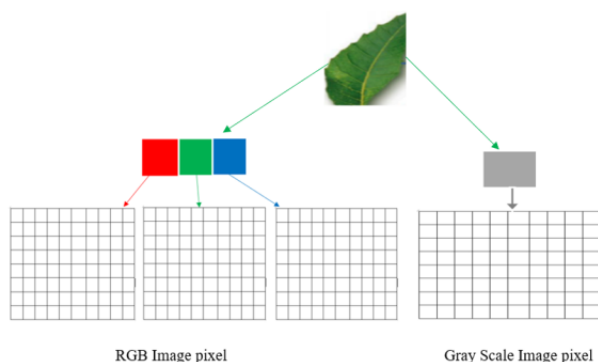


Figure 3.5: Difference between RGB and gray scale image

3.7 Feature Extraction

A fine-tuned architecture of CNN is used to extract useful leave features and feed them to the classifier. The fine-tuned models, Mobile Net, VGG16, and Inception_V3 are adopted. These feature extractions are processed in the base convolution part of the convolutional neural network. This is achieved with the help of filtering kernel (convolution), Max pooling, and ReLU activation function. In this feature extraction, the deep learning model extracts the necessary

features of the plant image automatically.

3.7.1 How CNN Extract Features

When going from the first layer to the last, CNN performs very complex tasks for feature extractions. Figure 3.6 shows how CNN extract leaf features. At 1st, it detects simply identifiable patterns like horizontal and vertical lines present in leaves. At the 2nd layer, it gives more information than 1st layer like, it can detect different corners on leaves as a result it tries to identify the shape of leaves. At the 3rd layer, it further computes more complex feature map identification like extracting differently structured veins on the surface of leaves. At the 4th layer, it becomes more powerful to exploit each tiny vein structure and while going to a very deep layer it can identify the ubiquitous structure. So, by going through this execution CNN can identify leaves unique features for classification.

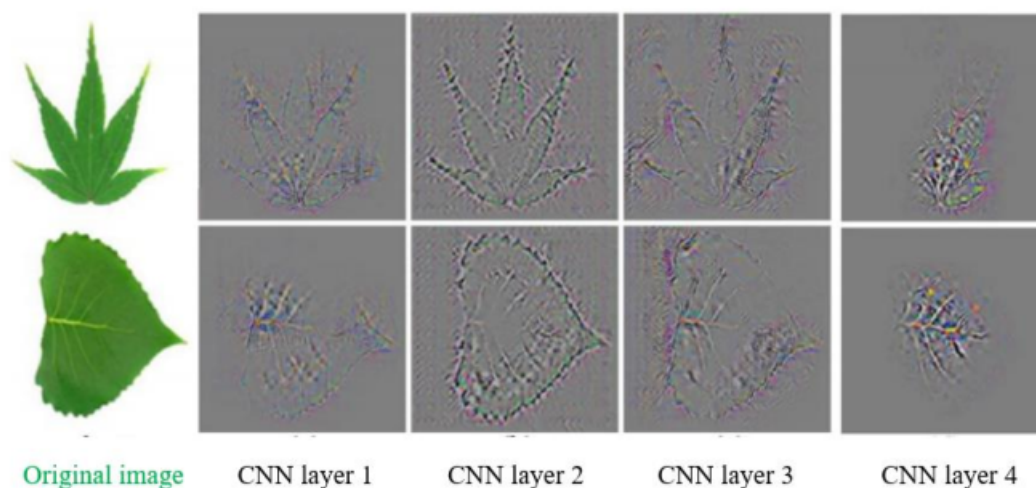


Figure 3.6: Feature extraction in the internal layers of CNN

3.8 Proposed CNN Model

Figure 3.7 shows the adopted architecture of CNN in this study. Figure shows, it has feature learning and classification layers. The feature learning layer is to extract plant leaf features and the classifier layer is to receive mapped features from feature learning layer and predict medicinal plant parts.

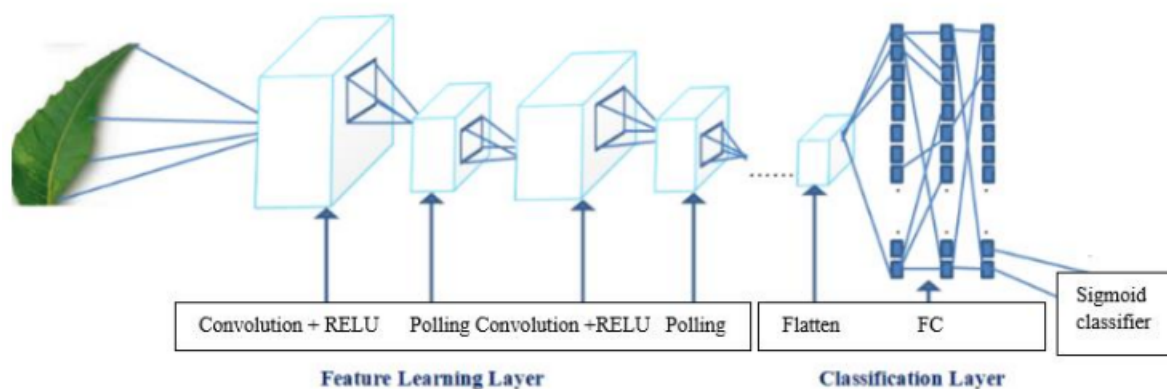


Figure 3.7: CNN architecture of the proposed model

3.8.1 Feature Learning

The learning (feature extraction) module called convolutional block, contains three operations, convolution, ReLU, and pooling. It used convolution to extract unique features from an input image of the medicinal plants using filters (kernels). ReLU makes the value of mapped (extracted) features positive by changing negative values of pixels to zero. Max pooling reduces the size of feature map produced by convolution to save the computational time. This part of the CNN also used this Max Pooling to filter the most effective feature descriptor values of input plant leaf images.

Convolution Processes

Figure 3.8 shows how convolution is performed in CNN. The figure shows the convolution process using input image of 5 x 5-pixel size, 1 kernel (filter) size of 3 x 3 with stride size 1. HS and VS represents horizontal shift and vertical shift respectively. The image is multiplied by the kernel to detect the edge, sharpen, blur, and perform other image processing tasks. Finally, the image filtered to 3 x 3 image size. So, in CNN using convolution, feature map can be extract from the input image as shown.

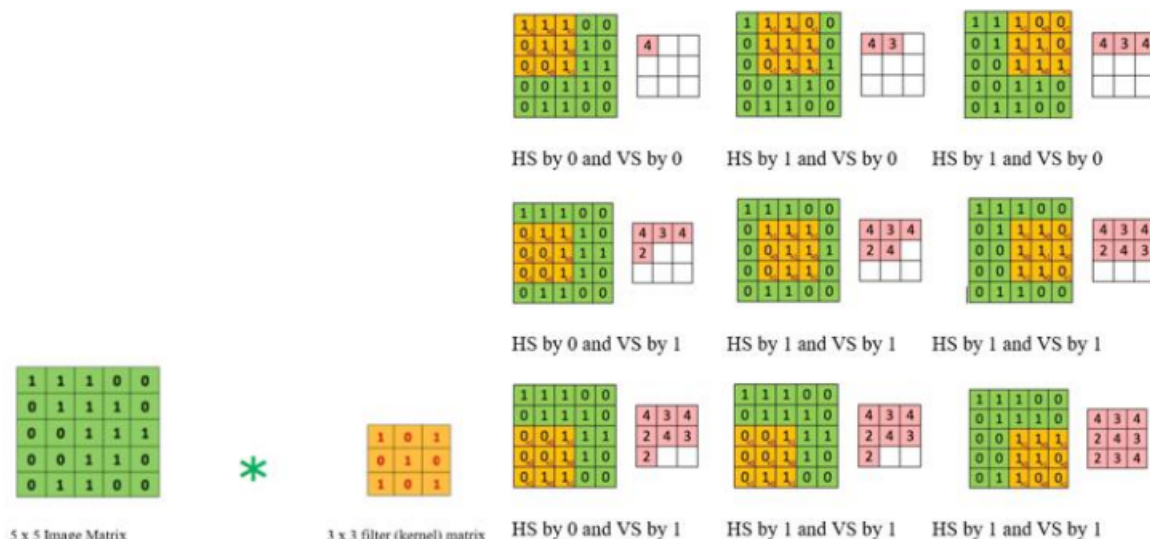


Figure 3.8: Convolution process of CNN

Polling

The purpose of polling in deep learning is, reducing the size feature map produced by the convolution process to reduce computational complexity. By how much size it reduces the feature map, depends on the window size of polling. If the size of a poll is 4x4, features map size will reduce by this size. There is a variety of polling such as Max Polling, average Polling, and mean polling. This study used Max polling to accelerate feature extraction from given input medicinal leaf image. With this technique, the highest pixel value is selected from a region depending on its size. In other words, max-pooling takes the largest value from the window of the image currently covered by the kernel.

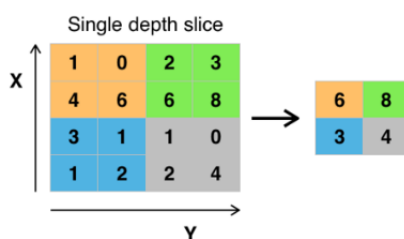


Figure 3.9: Max pooling

3.8.2 Classification

The classification part contains a flattened, fully connected neural network, dropout, and softmax layer. The flatten layer helps to change 2D image pixels value to 1D array and fit each of the the encoded image to a fully connected layer. At fully connected (dense) layers encoded image features are processed through each neuron in the layers. Dropout is applied to reduce over fitting. At the end, there is a Softmax activation function to compute the probability of output neurons between 0 and 1 for class determination. Figure 3.10 shows the general operations that are performed in this study. In the feature extraction table F1, F2, F3. . . Fn represents features, D1, D2, D3. . . Dn represents input image data which is feature vector as 2D array and C1, C2, C3. . . Cn represents target classes.

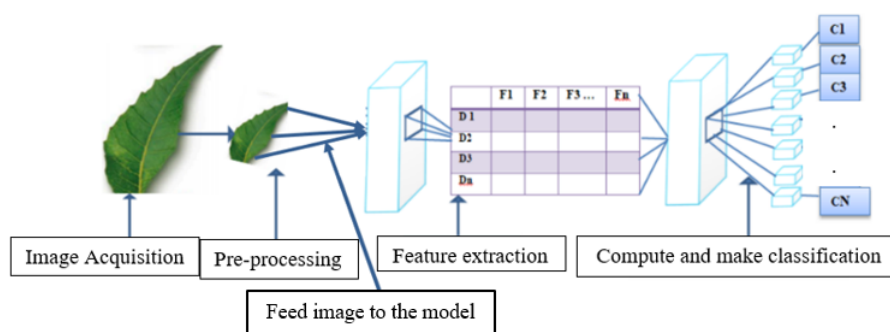


Figure 3.10: Operational components of the Model

3.9 Loss Function

The categorical cross-entropy loss function is used in this study. Because for the proposed study, the multi class classification is used. In multi-class classification the probability of each target label is calculated using a Softmax classifier and the one that has a higher probability than the remaining labels is predicted as the output class for a given specific input plant image.

3.10 Multi Class Setup

Medicinal plants have more than one function to cure human diseases based on their plant parts. This part of the plant takes as a target label for corresponding input leaf image to classify

it using multi class classification. the target class labeled as $L = [l_1, l_2, l_3, l_n]$ where L is the labels and taken as a target class for a given input image. The value of L_n is annotated either 0 or 1. For a specific input image $L_n=0$ means, the input leaf image does not contain that plant part (the label is plant part) and $L_n = 1$ means, input leaf image have that plant part.

3.11 Transfer Learning

In this technique the pretrained models are adapted when there is either of the three option such as, it there is the common data set, if the problems to be solve is the same, or if both data sets and problems are the same. This adaptation of pretrained models helps to obtain good model performance with small number of data, short period of time, and small number of training epochs. Transfer learning in short the process of tuning training and model hyperparameters of pretrained models. Figure 3.11 describes the possible options when using transfer learning.

1. The model train entirely when there is large data and different from pretrained models data set
2. The small number of convolutional base (feature extractor) layers are trained when there is large number of data and similar to pretrained models data set.
3. The majority layer of the base convolution trained when the data is small and different from pretrained models data set.
4. Make all layer of the convolutional base if the data is small and the same as the pretrained models data set. In this case only the classifier layers are tuned (modified).

In this study, transfer learning techniques are adopted to extract features, train the models, and predict medicinal plant parts to save computation resource and time. The adopted models are pre-trained with the image net. In this procedure, some layers of the models are tuned with the new weights and the remaining are left to freeze. At the fully connected layer, the new dense layers, drop out and batch normalization are added. The advantages from this transfer learning are, the training can be completed in a short period of time, no need of large number for training epochs, and also the data needed for training are small. Thus, to customize these

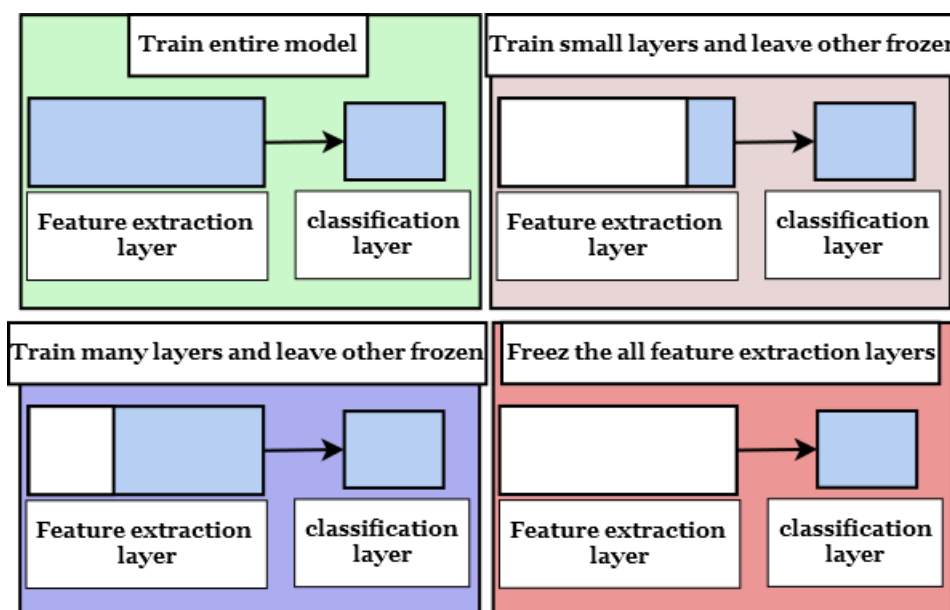


Figure 3.11: Alternatives in transfer learning technique

pretrained models in addition to layer tuning, the deep learning hyper parameters are tuned up on the models.

3.11.1 Mobile Net

The Mobile Net model is based on depth wise separable convolutions which is a form of factorized convolutions that factorize a standard convolution into a depth wise convolution and a 1x1 convolution called a point wise convolution. For Mobile Nets, the depth wise convolution applies a single filter to each input channel. The point wise convolution then applies a 1x1 convolution to combine the outputs of the depth wise convolution [33]. A standard convolution does both filters and combines inputs into a new set of outputs in one step. The depth wise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. The last 4 layers of convolutional base (feature learning part) is trained using the new weight and leave the remaining layer freeze which means no weight will be modified for these layers. At the fully connected layer (classification layer), four layers with a 1024 unit and ReLU activation functions are used. Finally, the softmax layer with 8 units are used because there are eight classes. Another interesting thing is, Batch Normalization is added to facilitate training rates with small epochs. Mobile Net is learned with 128 X 128 size of input image. The model has a total of 28 convolutional layers. However, since two fully connected

layers are added, it is changed to 30 convolutional layers.

3.11.2 VGG16

In this model, we fine-tuned some parts of the layers and weights. New weight on the last four layers of the convolutional base (the learning layer) are trained, the remaining layers are frozen and two fully connected layer added with 1024 units and ReLU activation function after removal of the top layer. Batch normalization layer is used for the same purpose as Mobile Net. In this model, a dropout layer with a value of 0.5 is added to overcome the problem of over fitting. All trainable layers, except the last four layers of the convolutional base, are frozen. FC units shrink from 4096 to 1024 and input shape is reduced from 224 X 224 to 128 X 128 pixels. This is done to save computational time and memory with the available local dataset.

3.12 Inception_V3

On Inception_V3 the same tuning mechanism applied as Mobile Net and VGG16 except in this model all base convolutional layers are frozen and dropout of 0.9 is used at FC.

3.13 Model Evaluation Metrics

Accuracy, precision, recall, and F1 score are used in this study. The macro, micro, and weighted average are also used for both precision, recall, and F1 score.

3.13.1 Precision

Precision calculates how precise/accurate our model is by computing how many of them are actually positive out of those predicted positive. Precision is a good measure to determine when the costs of false Positive is high. It is also a means of how often the model is correct when it predicts class as true positive.

$$Precision = \frac{TP}{TP + FP} \quad (3.1)$$

3.13.2 Recall

Recall actually calculates how many of the Actual true positives our model capture through labeling. Recall is the model metric used to select our best model when there is a high cost associated with False Negative.

$$Recall = \frac{TP}{TP + FN} \quad (3.2)$$

3.13.3 F1_score

F1 Score used to seek a balance between Precision and Recall. It is useful when there is an imbalanced data class distribution. Hence, this study give emphasis to this metric, because there are imbalanced classes.

$$F1_{score} = 2 * \frac{precision * recall}{precision + recall} \quad (3.3)$$

3.13.4 Micro-Average

Calculate metrics globally by counting the total true positives, false negatives, and false positives. This is also crucial in this study due to correctness on our imbalanced classes. In this study 'n' is 8 because there is 8 classes.

$$Precision_{micro} = \frac{\sum_{class=1}^n TPofclasses}{\sum_{class=1}^n TPofclass + FPofclass} \quad (3.4)$$

$$Recall_{micro} = \frac{\sum_{class=1}^n TPofclasses}{\sum_{class=1}^n TPofclass + FNofclass} \quad (3.5)$$

$$F1_{score}_{micro} = 2 * \frac{precision_{micro} * recall_{micro}}{precision_{micro} + recall_{micro}} \quad (3.6)$$

3.13.5 Macro-Average

Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account. So, it is not necessary for this study due to the presence of imbalanced classes, but for experimental investigation purpose, this metric is used.

$$Precision_{macro} = \sum_{class=1}^n \frac{precision\ of\ class}{number\ of\ class} \quad (3.7)$$

$$Recall_{macro} = \sum_{class=1}^n \frac{recall\ of\ class}{number\ of\ class} \quad (3.8)$$

$$F1_score_{macro} = \sum_{class=1}^n \frac{F1_score\ of\ class}{number\ of\ class} \quad (3.9)$$

3.13.6 Weighted Average

Calculate metrics for each label, and find their average weighted by support (the number of true instances for each label).

$$Precision_{weighted} = \sum_{class=1}^n weight * precision\ of\ class \quad (3.10)$$

$$Recall_{weighted} = \sum_{class=1}^n weight\ of\ class * recall\ of\ class \quad (3.11)$$

$$F1_score_{weighted} = \sum_{class=1}^n weight\ of\ class * F1_score\ of\ class \quad (3.12)$$

3.14 Prediction

To predict the type of a medicinal plant image, it is enough to load the trained model, read the image files, feeds to the model, and finally get the result as a percentage value for the specified leaf part.

3.15 Requirements

Table 3.2 describes the requirements that are used by this study to preprocess data, extract features, train and test the models. To handle this study, Google Colab is used by achieving the requirement mentioned in the table. This collaborator is one type of grid system provide by google that gives free 12 RAM with GPU (graphics processing unit).

Table 3.2: Requirements

Software	Language	Python 3.7
	Software tools	<p>Google Colab with the following package</p> <ul style="list-style-type: none"> • Jupyter note book • Tensor flow framework used as back end • Keras frame work run on the top of Tenser flow used as front end
Hardware	Computing GPU optimum of 16 GB	
	Processing speed optimum of 2.8 GH	
	Internal memory optimum of 12 Gb	

Chapter 4

Experimental Results and Discussions

4.1 Preliminary Note

This chapter discusses experimental results in detail. The image intensity variations are analyzed using image histogram. During experiment, data is splitting strategy conducted, results are analyzed and compared. The effect of different hyper-parameters on the performance of models are analyzed by tuning it dynamically. The performances of the models in both training and testing datasets are analyzed and discussed including the corresponding hyper-parameters. And also results of evaluation techniques applied on models are discussed. At the end samples of the predicted leaf results are incorporated.

4.2 Dataset Variation Visualizations Using Histograms

The main purpose of incorporating this image histogram is, only for personal understanding of RGB image variations present in the locally collected dataset. it is possible to understand the frequency of pixels in RGB channels, brightness and, darkness of the images by simply observing the vertical, left, and the right parts of the histogram.

4.2.1 Light Intensity Variation

In this study images are captured in different environment and camera. This creates variety of data set based on intensity strength. Hence, data augmentation is not proceed during training



Figure 4.1: Intensity variation in the dataset

phase. Figure 4.1, shows images of one leaf but in different pixel intensity and the histogram plot describes how it varies based on its three-channel pixel intensity and distribution through out the different images.

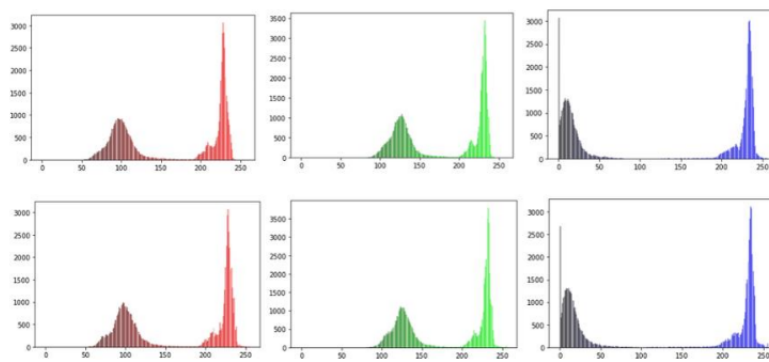


Figure 4.2: Intensity level of sample 1

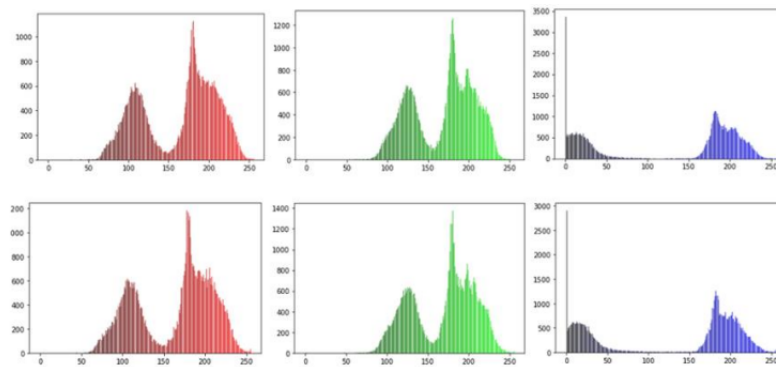


Figure 4.3: Intensity level of sample 3

Figure 4.3 shows the histogram of S3 image. It is different from sample one (S1). The leaf S1 and S3 are the same but this variation is due to light intensity while capturing the image. This variation of the datasets in a single leaf makes the model to classify consistently while it faces different images captured by different cameras and environments.

4.3 Training and Testing Dataets

After conducting annotation and preprocessing, training and testing split applied for model evaluation. In this study, the data is split 80% for training and 20% for test. 15,100 total medicinal plant images are prepared. From this data first 20% taken for testing which is 3,020 images and 12080 remains for testing, and further from these data again 20% (2,416 images) is taken for validation. So, the data are split in to 9,664 for training, 3,020 for testing (unseen data during training), and 2,416 for validation.

4.4 Hyper-parameter Tuning

The parameters on the selected pretrained models are fine-tuned to make it suitable as much as possible compatible with local data. The following table illustrates about the hyperparameters used in this study to fine-tune the CNN architectures. In all models, weight of image net and custom weight are used and the dense layer removed and replaced with new convolutional layers including batch normalization and dropout layer. Batch normalization to increase the speed of learning and acquire higher accuracy values.

Table 4.1: Hyperparameter tuning values

Hyper-parameters		Mobile Net	VGG16	Inception_V3
Batch size	32	Applied	Applied	Applied
	64	»...»	»...»	»...»
	128	»...»	»...»	»...»
Learning rate	1e ⁻⁴	»...»	»...»	»...»
	1e ⁻³	»...»	»...»	»...»
Optimizers	Adam	»...»	»...»	»...»
	Adamax	»...»	»...»	»...»
	Adagarad	»...»	»...»	»...»
Dropout	0.5,0.5,0.9	»...»	»...»	»...»
Weight modifica- tion	At base convolution	»...»	»...»	Not used

4.5 Training and Testing Models

4.5.1 Training and Testing Accuracy

The performance result is described in both graph and table forms. For training, iterations (number of epochs) 30 is used for all type of hyper-parameters and tested models. Because using epoch 30, optimal results are obtained from three models. Beyond this iteration the training accuracy remains the same. there is no change at all in training accuracy and loss. For the trained models in each experiment, the best optimizer are selected and mentioned in the table with the corresponding batch sizes. In this study more than 54 experiments are conducted. But only the preferred results are described here while the remaining and detail experimental results are shown in the appendix.

4.5.2 Mobile Net

Experiment One

Table 4.2: Mobile Net training and testing accuracy using LR $1e^{-4}$ with different BS

Batch size	Optimizers	Training accuracy	Testing accuracy	Training loss	Testing loss
32	Adamax	99.84	99.44	0.86	2.76
64	Adam	99.78	99.24	0.93	3.58
128	Adam	99.78	99.27	0.59	2.44

In table 4.2, Adamax gives higher values on both training and testing accuracy. The table indicates training accuracy of 99.84% and testing accuracy of 99.44% are achieved. Hence using a batch size of 32 with Adamax optimizer is acceptable. Using batch size 64 with Adam optimizer also gives recommended performance. Figure 4.4 shows the best case of training and validation accuracy/loss of fine-tuned Mobile Net model using Adamax optimizer, batch size of 32. When the training and testing losses are analyzed, Adam obtains a minimum loss than Adamax. So, it is possible to deduce that Adam also a preferable optimizer for the locally collected dataset. Generally Mobile Net gives a higher result with these specified batch sizes and optimizers. The remaining graph representation is plotted in appendix A.1.

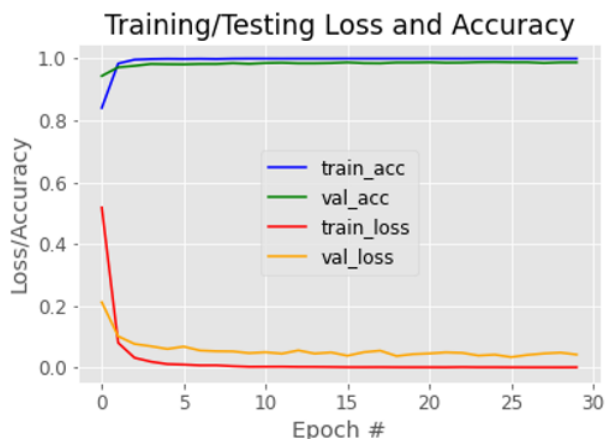


Figure 4.4: Accuracy and loss of Mobile Net for training and testing dataset in the best case

Experiment Two

Table 4.3: Mobile Net training and testing accuracy using LR $1e^{-3}$ with different BS

Batch size	Optimizers	Training accuracy	Testing accuracy	Training loss	Testing loss
32	Adamax	99.72	98.97	1.30	4.49
64	Adamax	99.81	99.17	0.74	2.70
128	Adam	99.82	99.50	0.46	1.83

These are the result of using a learning rate of $1e^{-3}$. On a batch size 32, Adamax gives higher accuracy. But it causes the model to become overfitted. So, it is recommended to use Adam using batch size 64 and 128. Generally, Adam has optimum performance using batch size 128. Because the training and testing losses using these optimizer and batch size are small compared to the other. The graph representations are described in appendix A.2.

4.5.3 VGG16

Experiment One

For this model, Adam using the batch size of 64 and 128 gives good accuracy. When compared among loss values, Adam using a batch size of 128 is give a preferable result.

Table 4.4: VGG16 training and testing accuracy using LR $1e^{-4}$ with different BS

Batch size	Optimizers	Training accuracy	Testing accuracy	Training loss	Testing loss
32	Adam	99.74	99.07	1.15	4.87
64	Adam	99.79	99.30	0.94	3.19
128	Adam	99.78	99.37	1.04	2.90

Experiment Two

Table 4.5: VGG16 training and testing accuracy using LR $1e^{-3}$ with different BS

Batch size	Optimizers	Training accuracy	Testing accuracy	Training loss	Testing loss
32	Adagrad	99.64	98.64	1.29	4.53
64	Adamax	99.69	98.41	1.69	8.83
128	Adamax	99.21	96.99	3.73	14.16

Table 4.5 shows, Adagrad using batch sizes of 32 and Adamax using 64 achieve good performance. The graphical representations are shown in Appendix B.2. When looking at the table, the model with Adamx using a batch size of 128 is overfitted. The comparison of testing loss in batch size 32 and 64 shows, a testing loss using a batch size of 32 is small. Hence, for this experiment, Adagrad is selected as the optimum optimizer.

4.5.4 Inception_V3

Experiment One

Table 4.6: InceptionV3 training and testing accuracy using LR $1e^{-4}$ with different BS

Batch size	Optimizers	Training accuracy	Testing accuracy	Training loss	Testing loss
32	Adam	96.12	90.53	15.49	28.39
64	Adam	95.58	91.26	15.63	26.75
128	Adam	95.32	90.03	15.43	29.48

Table 4.6 shows inception_V3 performs with a minimum training and testing accuracy compared to Mobile Net and VGG16. But when compared with itself Adam gives good results

using the three batch sizes. So Adam is a preferable optimizer while training the Inception_V3 model for local datasets. Experiment two is shown in Appendix D.3.

4.6 Comparisons Among Fine-tuned Models

In figure 4.5 and 4.6, the study shows the experimental results of testing accuracy and testing losses. Because the main thing during the training of a model is, how much it performs when it is tested by the new (unseen) data. The testing loss also highly determines the efficiency of the models. The training accuracy and losses are mentioned in the Appendix part. The vertical axis represents accuracy and loss percentage in accuracy and the loss figures respectively. The horizontal axis is a hyper-parameter called optimizers.

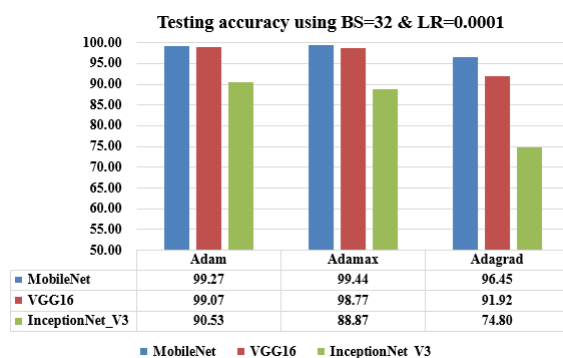


Figure 4.5: Testing accuracy of the models using BS = 32 and LR = 1e⁻⁴

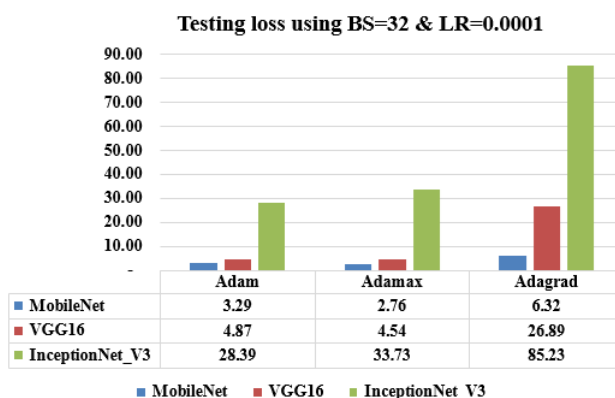


Figure 4.6: Testing Loss of the models using BS = 32 and LR = 1e⁻⁴

Figure 4.5 and 4.6 shows as the models are obtained desirable performances in both testing and loss results. This result is obtained by using batch size 32 and learning rate 1e⁻⁴. The graph

shows Mobile Net has scored a good performance compared to VGG16 and Inception_V3. The figure shows, Mobile Net performs 99.44% and 99.27% using Adamax and Adam respectively for the testing datasets and 2.76 % of testing loss. This is a good result when compared to the remaining models. Hence from optimizer Adamax and Adam, from learning rate $1e^{-4}$, and from Model Mobile Net are selected for the classification of plant parts. Because it has a majority rank when compared to other hyper-parameters and models based on 54 experimental results.

4.7 Classification Report

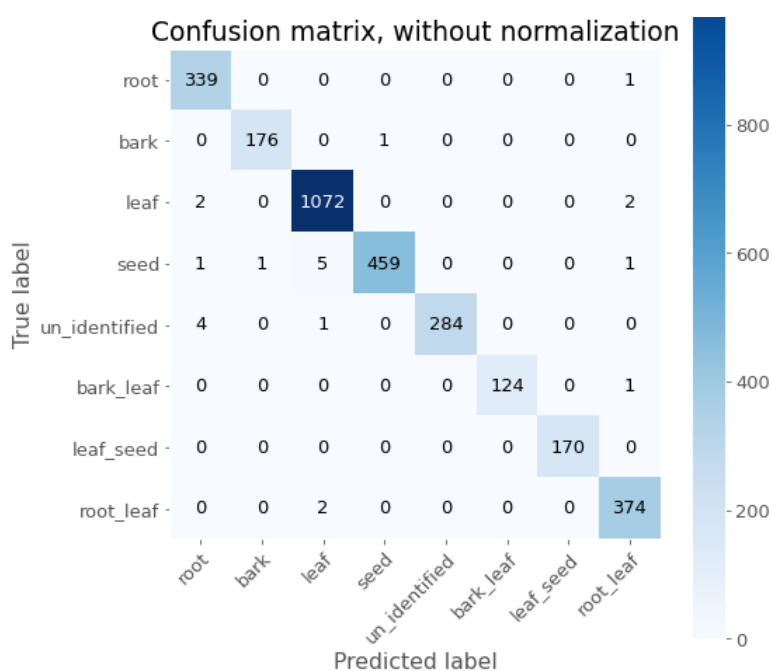


Figure 4.7: Confusion matrix obtained from Mobile Net

Figure 4.7 represents the confusion matrix on which the models are measured with the test data set by observing how many test samples of the data are classified correctly and how many are misclassified in each class (category). This classification report is taken as the best performance from the 54 experiments. The result is obtained using a learning rate of $1e^{-4}$, batch size of 32, and optimizer of Adamax. Table 4.7 shows how many of the sample data are classified in true positive (TP), false positive (FP), true negative (TN), and false-negative (FN) in each of the Eight classes. These values in the table are taken from a confusion matrix. This is to clarify what is done by the models. The root of the plant is taken as an illustration for the confusion

matrix. It has the true positive value of 339, false-positive value 1, true negative 2,673, and false negative 7. Hence the root has a total test data of 340. From these data, 339 samples are classified into the root category. Only 1 sample is classified as root_leaf. For further clarification, leaf_seed can be taken as an additional observation. There are 170 samples labeled in leaf_seed and the model classifies all these samples correctly without any misclassification. These shows a desirable result of the classification task handled by Mobile Net.

Table 4.7: Analysis of correctly classified and misclassified classes from the confusion matrix

Classes	True positive	False positive	True negative	False negative
Root	339	1	2673	7
Bark	176	1	2842	1
Leaf	1072	4	1936	8
Seed	459	8	2552	1
Un_identified	284	5	2731	0
Bark_leaf	124	1	2895	0
Leaf_seed	170	0	2850	0
Root_leaf	374	2	2639	5

Table 4.8 is taken from the classification report of Mobile Net to show the performances of the model in each category (class). The results of F1_score is taken for the description because this evaluation metric takes the advantages of both precision and recall by making the harmonic mean of Precision and Recall. The F1_score shows, the model obtained the desirable performances on both classes, especially on two classes. For these two classes, the model becomes 100% accurate. It is also possible to make sure by referring from the above table. This table shows as the model performs with 0 and 1 misclassification for leaf_seed and bark_leaf categories respectively. The prediction results at the end of this chapter also another evidence for the performance of a model that is adopted in this study..

Table 4.8: Performance of Mobile Net on individual classes

Classes	Precision in %	Recall in %	F1_score in %	No. of test data
Root	98	100	99	340
Bark	99	99	99	177
Leaf	99	100	99	1076
Seed	100	98	99	467
Un_identified	100	98	99	177
Bark_leaf	100	99	100	125
Leaf_seed	100	100	100	170
Root_leaf	99	100	99	376
Micro-average	99.39	99.25	99.32	3020
Micro-average	99.27	99.27	99.27	
Micro-average	99.28	99.28	99.28	

4.8 Model Evaluations

All the above results are taken from the performance result of the model. But there are no balanced classes to evaluate the model only with this accuracy because accuracy becomes a valid measurement of given model performance when the model is trained with balanced datasets. Accuracy simply evaluates how much percent a model has learned from the input data. It does not consider model performance with respect to individual classes. So, other metrics are used to evaluate all trained models, such as precision, recall, and F measure. Micro averaged, macro averaged, and the weighted average for all metrics (precision, recall, and F1_score) also implemented in this study. Detailed results of these metrics are described in appendixes.

4.8.1 F1 Measurement

Most of the time when there are imbalanced data distribution in each classes, F1_score is the best metric to evaluate the models. It is obvious that all researchers used this metric for their model evaluation with imbalanced data. By applying F1_score, the worst case and the best one are identified among different models. Figure 4.8 indicates Mobile Net performs best using a batch size of 32 and an optimizer of Adam and Adamax. Further Adamax makes the

model more accurate than using Adam. This indicates that Mobile Net is well trained on local medicinal plant data than other models. The prediction result of this model is discussed in the last section of this chapter. In figure 4.8, the vertical axis indicates a micro average value in percentage and the horizontal axis is a hyper-parameters. So far a micro average of F1_score,

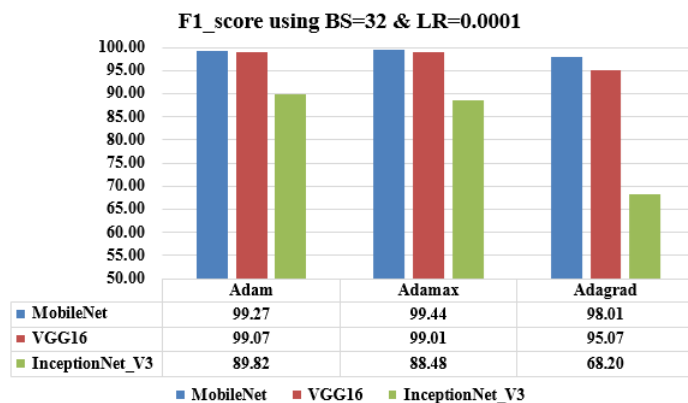


Figure 4.8: Overall F1_score for tested models

the performance of the models are seen using different learning rates and batch size. Recall and precision are described in the appendix D.1 and D.2 using tables.

4.9 Experiment without Batch Normalization

In earlier sections, the empirical results using the addition of batch normalization at fully connected layers have been seen. In this section, analyzation of how the fine-tuned models perform without batch normalization is discussed. Only a learning rate of $1e^{-4}$ and a batch size of 32 are taken because we needed to understand the effect of the absence of batch normalization. Other hyper-parameters are also tried but did not achieve a good result. The optimizer is the same as in previous experiments. Figure 4.9 and 4.10, shows experimental results of all fine-tuned models accuracy and loss for testing data sets respectively. This decreases the accuracy of models and increases the testing loss.

Table 4.9 shows by how much percent the models decrease its test accuracy and increase its loss when it is trained without batch normalization at the FC layer. This result is obtained by subtracting the values of the performances with the absence of batch normalization. The average values of the decreasing of testing accuracy and increasing of testing loss are due to the absence of batch normalization at fully connected layers of the CNN. Batch normalization

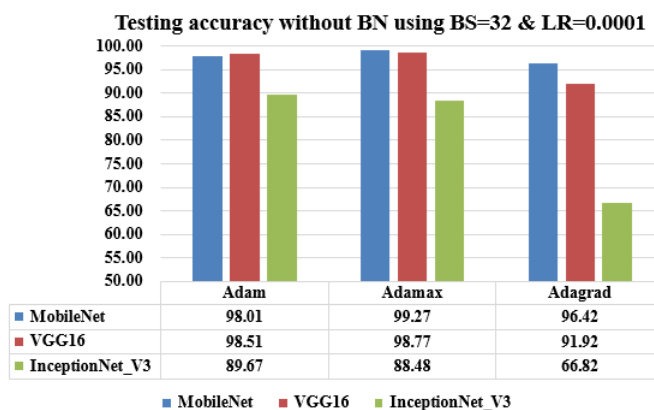


Figure 4.9: Testing Accuracy of the models without BN

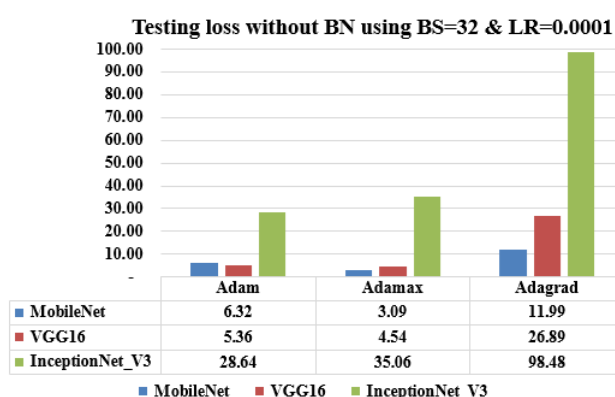


Figure 4.10: Testing loss of the models without BN

makes each individual convolutional layer in CNN to learn from image features independently and also avoid covariance shift. So, the absence of this advantages in CNN layers causes a model to decrease its accuracy and increase with its testing loss.

Table 4.9: Performance difference between the absence and the presence of batch normalization in FC layer of the models

Models	Test accuracy without BN				Test loss without BN			
	Adam	Adamax	Adagrad	Average	Adam	Adamax	Adagrad	Average
MobileNet	-1.26	-0.17	-0.03	-0.49	+3.03	+0.33	+5.67	+3.01
VGG16	-0.56	0	0	-0.56	+0.49	0	0	+0.49
Inception_V3	-0.86	-0.39	-7.82	-5.64	+0.25	+1.25	+13.25	+4.94

Generally, when looking at the whole hyper-parameters and models of the experiments, table 4.10 gives a brief analysis of the experimental result of this study. From table 4.10 it

is necessary to understand that for the classification of a medicinal plant part, the parameters mentioned in the best case is advisable. The hyper-parameters and model mentioned in the right side of the table makes the model perform low when compared to the best case. This comparison is made based on the values obtained from the model evaluation metric called F1_score.

Table 4.10: Model and hyperparameters selection summary

Hyper-parameters	Best case	Worst case accuracy
Learning rate	$1e^{-4}$	$1e^{-3}$
Optimizer	Adam and Adamax	Adagrad
Batch size	32	128
Models	MobileNet	Inception_v3

Finally, this study obtained that Mobile Net is the fastest model to train and test. It scores higher performance and is suitable to classify the medicinal plant part. The reason is not due to the small number of the hidden layers, instead mobile Net used a depth wise separable convolutional layer. Depth wise separable convolution reduces the total scalar multiplication produced by convolution process.

4.10 Computational Time Variation Among Models

As discussed, different models are trained and tested. The following table describes the execution time of each model. This comparison is performed based on a learning rate of $1e^{-4}$, an optimizer of Adam, and different batch sizes. These computation times are taken from experimental results. The goal is to analyze the training, testing, and prediction speed of the fine-tuned models with a new dataset. The experiment finds, when the batch size increases execution time decreases.

Table 4.11: Computational time variance among models

Models	Batch size	Training time	Testing time	Prediction time
Mobile Net	32	3 min 28 s	6.44 s	5.77 s
	64	2 min 51 s	6.4 s	5.72 s
	128	2 min 38 s	6.28 s	5.50 s
VGG16	32	9 min 35 s	19.3 s	3.24 s
	64	8 min 11 s	19.3 s	3.31 s
	128	8 min 19 s	19.3 s	3.29 s
Inception_V3	32	5 min 28 s	12.4 s	1.32 s
	64	4 min 30 s	12.3 s	1.76 s
	128	4 min	12.2 s	1.23 s

4.11 Predictions

The new images are prepared and loaded into the saved model. Finally, the result is analyzed by displaying the plant part and the percentage of each part on the top of the leaf image. This prediction is handled by fine-tuned models that perform higher efficiency which is Mobile Net using a learning rate of $1e^{-4}$, batch size of 32, and optimizer Adamax. From optimizers Adam also gives almost good prediction results. This implies that a Mobile Net model is best suitable for the plant part classification.

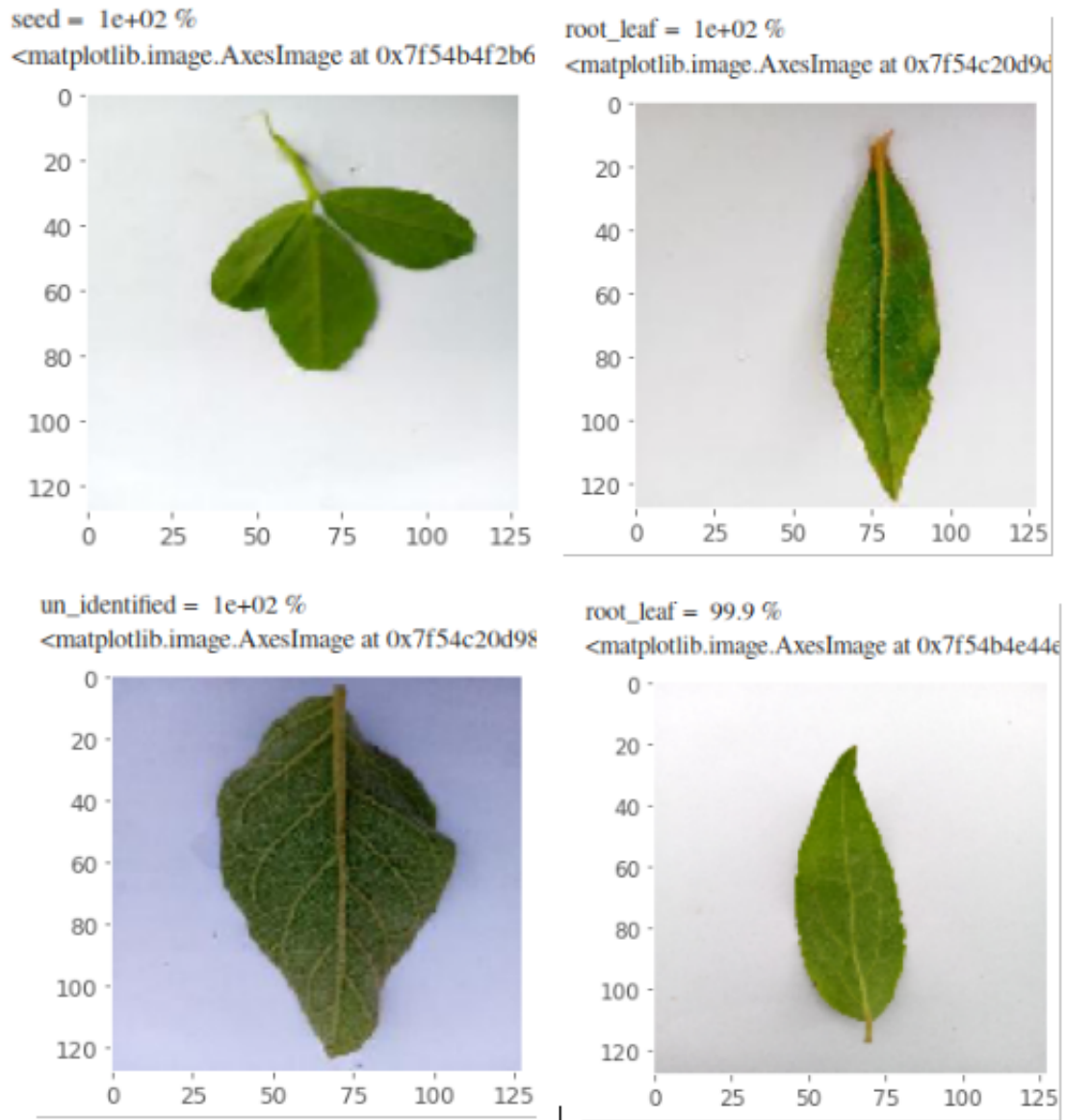


Figure 4.11: Prediction results of sample medicinal plant using Mobile Net

Chapter 5

Conclusion and Recommendations

5.1 Conclusion

Different researchers proved that plants have a vital role in the curing of human diseases. And they proved plants are the direct sources of medicine. Its root, bark, leaf, and seed can be used to prepare the medicines. As understood from data collections, leave part of the plants takes the majority role in curing diseases. Experiments using CNN have shown superior accuracy in image classification but to accurately train the models based on this neural network, different hyper-parameters should be tuned on the dataset being used. This research is the first to classify the specific parts of medicinal plants using multi-class classification. Since there no published medicinal plants data based on their specific parts, the dataset is prepared based on the reference of pathobiological researches and consultation of experts who are currently working on preparations of traditional medicine.

From experimental results, we can conclude that over fitting of the models does not only depend on the number of a data set and the depth of hidden layers in CNN. But is also affected by the total number of scalar multiplication of filters (kernel) calculated by the convolution process. In this study, Mobile Net contains 30 convolutional layers and VGG16 has only 16 layers. But the Mobile Net does not overfit than VGG16. Because Mobile Net used depth- wise separable convolution technique. This technique reduces the scalar multiplication of the kernel by a factor of 1/20. So, the total number of scalar multiplication calculated in Mobile Net is

very small compared to VGG16. That is why Mobile Net does not over fit even though it has the maximum number of convolutional layers than VGG16.

From the result of this study, three main pillar points are identified. 1st, batch normalization at the fully connected layers of the CNN makes the model learn fast in a small number of training epochs. 2nd, utilization of plant leaves image from its backside enables the models to extract the vein features accurately. The combined effect of the two cases leads the model to score a good performance while it is tested by unseen (new) data. 3rd, depth wise separable convolution is highly determine the over fitting and under fitting of the models. Finally, this study deduces that using a learning rate of 1e-4, batch size of 32, and optimizer of Adam and Adamax, the models achieved improved performances of 99.44%, 99.01%, and 88.48% for Mobile Net, VGG16, and Inception_V3 respectively. This result is based on F1_score performance metric. Generally, after conducting more than 54 experiments on 15,100 images of medicinal plants, the study concludes Mobile Net is the fastest model to train and test medicinal plant parts. This model is also selected as suitable deep learning technique to solve the identified problem by achieving higher classification performance than VGG16 and Inception_V3.

5.2 Recommendations

This study introduces the idea and testing of the identification and classification of medicinal plant parts using deep learning technique based on multi class categories. This is to answer the question, which part of medicinal parts used to medicine preparation?. But the study does not incorporate the identification of diseases cured by the identified parts of the plant, limited with small number of data sets, and manually find the optimal learning rate using grid search method. For the future work, we recommend the researchers to incorporate the following concept for more relevant results.

- It is good using FastAI (a deep learning library running on top of Pytorch) to make the classification task easier and fast by finding the optimal learning rate using learning rate finder.
- Increasing the number of data set will lead the model obtain accurate results.
- If researchers will incorporate the corresponding diseases to be cured by specified parts, the output of a system will be more useful and relevant.
- Multi label classification is suitable than multi class classification for the data employed in this study. But multi label has lower performance than multi class mechanism. If the researcher works on the way that can increase the model performance, using multi label classification makes the research finding will be appreciable.

References

- [1] Ashenif Tadele, "Ethiopian Herbal Medicine Research Article Profile," Traditional and Modern Medicine Research Directorate, September, 2017.
- [2] Misganaw Meragiaw, Zemedede Asfaw, "Review of Antimalarial, Pesticidal and Repellent Plants in The Ethiopian Traditional Herbal Medicine," Journal of Herbal Science, Volume 3, Issue 13 April 2015.
- [3] Ermias Lulekal , Zemedede Asfaw, Ensermu Kelbessa and Patrick Van Damme, "Ethnomedicinal study of plants used for human ailments in Ankober District, North Shewa Zone, Amhara Region, Ethiopia," Ethnobiology Ethnomedicine (2013).
- [4] Mirutse Giday, Zemedede Asfaw , Thomas Elmqvist, Zerihun Woldu, "An ethnobotanical study of medicinal plants used by the Zay people in Ethiopia," Journal of Ethnopharmacology, Volume 85, Issue 1, March 2003.
- [5] Mirutse Giday, Zemedede Asfaw, Zerihun Woldu, "Medicinal plants of the Meinit ethnic group of Ethiopia: An ethnobotanical study," Journal of Ethnopharmacology, Volume 124, Issue 3, 30 July 2009.
- [6] Shipra Shah, Jahangeer A. Bhat, "Ethnomedicinal knowledge of indigenous communities and pharmaceutical potential of rain forest ecosystems in Fiji Islands," Journal of Integrative Medicine, Volume 17, Issue 4, July 2019.
- [7] Rajani S, Veena M.N, "Study on identification and classification of medicinal plants," International Journal of Advances in Science Engineering and Technology, Vol-6, Iss-2, Spl. Issue-2 Jun.-2018.

- [8] Naresh Y.G. and Nagendraswamy H.S., "Classification of Medicinal Plants: An Approach using Modified LBP with Symbolic representation," *Neurocomputing*, Volume 173, 15 January 2016.
- [9] Peffers, Ken, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. "A design science research methodology for information systems research." *Journal of management information systems* 24, no. 3 (2007): 45-77.
- [10] Ashenf Chekole, *An ethnobotanical study of plants used in traditional medicine and as wild foods in and around tara gedam and amba remnant forests in libo kemkem wereda, south gonder zone, amhara region, Ethiopia*, Addis Ababa University, 2011.
- [11] Phil Kim, *MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence*, Seoul, Soul-t'ukpyolsi, Korea, 2017, pp.1- 121.
- [12] Manohar Swamynathan, *Mastering machine learning with Python in six steps*, Bangalore, Karnataka, India, 2017, pp. 53 353.
- [13] Sunila Gollapudi, *Practical machine learning*, Birmingham B3 2PB, UK, 2016, pp.376.
- [14] Xiaodong Z, Xiaojie Wang, "Leaf Vein Extraction Based on Gray-scale Morphology," *I.J. Image, Graphics and Signal Processing*, vol 2.
- [15] D Venkataraman, Mangayarkarasi N, "Computer Vision Based Feature Extraction of Leaves for Identification of Medicinal Values of Plants," *IEEE International Conference on Computational Intelligence and Computing Research*. 15-17 Dec. 2016.
- [16] R.janani, A.Gopal, "Identification of selected medicinal plant leaves using image features and ANN," *international conference on advanced electronics system*, IEEE 11 November 2013.
- [17] Sivaranjani.C, Lekshmi Kalinathan and Amutha.R, "Real-Time Identification of Medicinal Plants using Machine Learning Techniques," *Second International Conference on Computational Intelligence in Data Science (ICCIDS)*. IEEE 21-23 Feb. 2019.
- [18] Luciano D. S. Pacifico, Larissa F. S. Britto, Emilia G. Oliveira and Teresa B. Luder-mir, "Automatic Classification of Medicinal Plant Species Based on Color and Texture

- Features," 8th Brazilian Conference on Intelligent Systems (BRACIS). IEEE 15-18 Oct. 2019.
- [19] Madhuri Bandara and Lochandaka Ranathunga, "Texture Dominant Approach for Identifying Ayurveda Herbal Species using Flowers," Moratuwa Engineering Research Conference (MERCon). IEEE 3-5 July 2019.
- [20] Shitala Prasad and Pankaj P Singh, "Medicinal Plant Leaf Information Extraction Using Deep Features," TENCON 2017 - 2017 IEEE Region 10 Conference, November 5-8, 2017.
- [21] Sue Han Lee a, Chee Seng Chan, Simon Joseph Mayo and Paolo Remagnino, "How deep learning extracts and learns leaf features for plant classification," Pattern Recognition, Volume 71, November 2017, www.elsevier.com/locate/patcog.
- [22] Jing Wei Tan, Siow-Wee Chang, Sameem Abdul-Kareem, Hwa Jen Yap and Kien-Thai Yon, "Deep Learning for Plant Species Classification using Leaf Vein Morphometric," IEEE/ACM Transactions on Computational Biology and Bioinformatics (Early Access), 19 June 2018
- [23] Dileep M.R and Pournami P.N, AyurLeaf: "A Deep Learning Approach for Classification of Medicinal Plants," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), IEEE 17-20 Oct. 2019.
- [24] C.Amuthalingeswaran, Mr. M.Sivakumar, Dr. P.Renuga, S.Alexpandi, J.Elamathi and S.Santhana Hari, "Identification of medicinal plant and their usage by using deep learning," 3rd International Conference on Trends in Electronics and Informatics (ICOEI), IEEE 23-25 April 2019.
- [25] Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall, "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning," arXiv preprint arXiv:1811.03378, 8, NOV 2018.
- [26] Abien Fred M. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," arXiv:18.3.08375v2 [cs.NE], 7 Feb 2019.

- [27] Sun S, Cao Z, Zhu H, Zhao J. "A survey of optimization methods from a machine learning perspective," IEEE transactions on cybernetics. 2019 Nov 18.
- [28] Diederik P. Kingma, Jimmy Lei Ba, "Adam: A method of stochastic optimization," arXiv:1412.6980v9 [cs.LG] 30 Jan 2017.
- [29] Sun S, Cao Z, Zhu H, Zhao J. "A survey of optimization methods from a machine learning perspective," IEEE transactions on cybernetics. 2019 Nov 18.
- [30] Hoffer E, Hubara I, Soudry D. "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," In Advances in Neural Information Processing Systems, 2017.
- [31] Ioffe S, Szegedy C. "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167. 2015 Feb 11.
- [32] Hoffer E, Hubara I, Soudry D. "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," In Advances in Neural Information Processing Systems 2017.
- [33] Amuthalingeswaran C, Sivakumar M, Renuga P, Alexpandi S, Elamathi J, Hari SS. "Identification of Medicinal Plant's and Their Usage by Using Deep Learning," IEEE, 2019 Apr 23.
- [34] Agarwal A, Negahban S, Wainwright MJ. "A simple way to prevent neural networks from overfitting," Ann. Stat. 2012;40(2).
- [35] Persson, Siri, "Application of the German Traffic Sign Recognition Benchmark on the VGG16 network using transfer learning and bottleneck features in Keras," 2018.
- [36] Li X, Zhao F, Guo Y. "Multi-label Image Classification with A Probabilistic Label Enhancement Model," In UAI 2014 Jul 23 (Vol. 1, No. 2).
- [37] Kundalia K, Patel Y, Shah M. "Multi-label movie genre detection from a movie poster using knowledge transfer learning," Augmented Human Research. 2020 Dec 1.

- [38] Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. "Mobile nets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861. 2017 Apr 17.
- [39] Zhang, Jeffrey and Tung, Albert, "Multiple Label Classification for Amazon Rainforest Images," Transfer, 2019.
- [40] <https://www.datacamp.com/community/tutorials/>, convolutional neural network using python, Retrieved in 28,12,2019 G.C.
- [41] 1 <https://medium.com/@phidaouss/> convolutional neural networks (CNN) or convnetsd7c688b0a207.Retrieved in 10,01,2020 G.C.
- [42] <https://www.kdnuggets.com/2019/12/convert-rgb-image-grayscale.html>, Retrieved on 05,10,2020 G.C.
- [43] <https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9>, Retrieved, 30,9,2020 G.C.
- [44] <https://datascience.stackexchange.com/questions/66388/what-is-the-formula-to-calculate-the-precision-recall-f-measure-with-macro-mi>. Retrieved, 30,9,2020 G.C.

Appendix A

Fine-tuned Mobile Net

In this section of this document, there is Experiment one and Experiment two. This is using learning rate $1e^{-4}$ and $1e^{-3}$ respectively. The following two boxes represent accuracy/loss graph position for training and testing with indicated optimizers inside the boxes. This is applicable to the graphs present in Appendix A and B



Figure A.1: Position index for screen shoot images

A.1 Learning Rate 0.0001

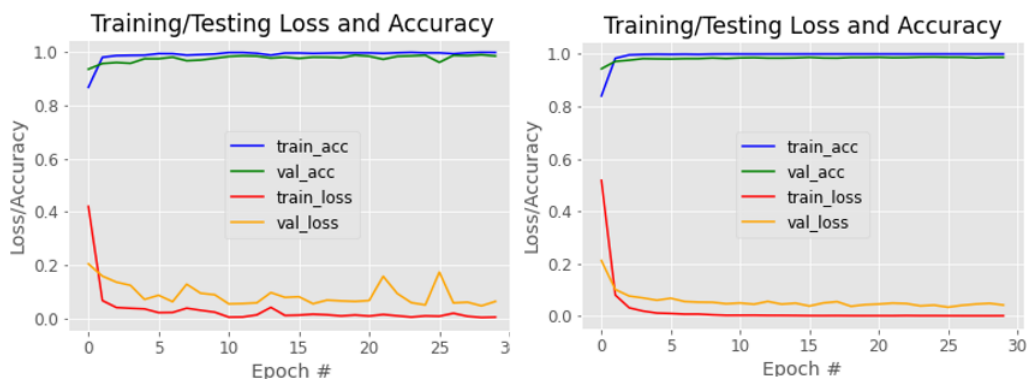


Figure A.2: Mobile Net accuracy/loss graph using batch size 32 and learning rate $1e^{-4}$

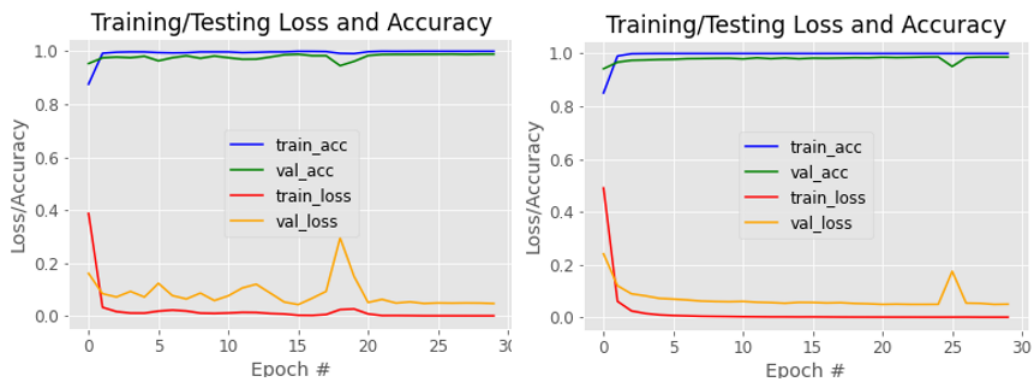


Figure A.3: Mobile Net accuracy/Loss using batch size 64 and learning rate $1e^{-4}$

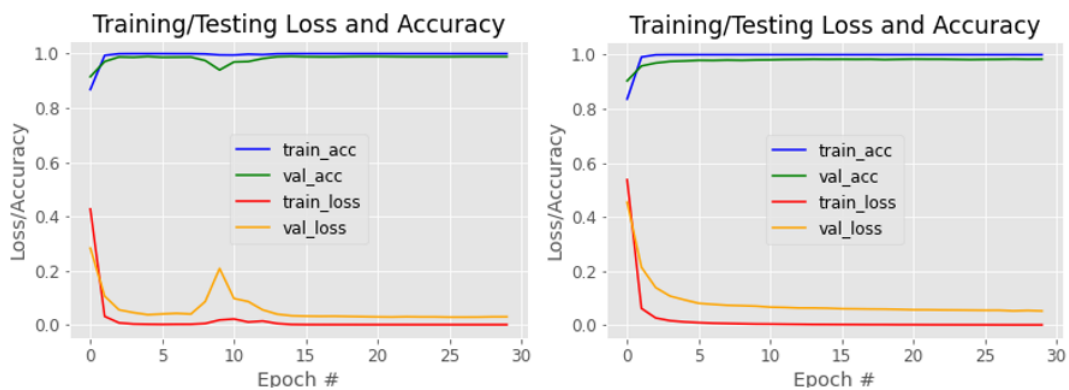


Figure A.4: Mobile Net accuracy/loss using batch size 128 and learning rate $1e^{-4}$

Appendix B

VGG16

B.1 Learning Rate 0.0001

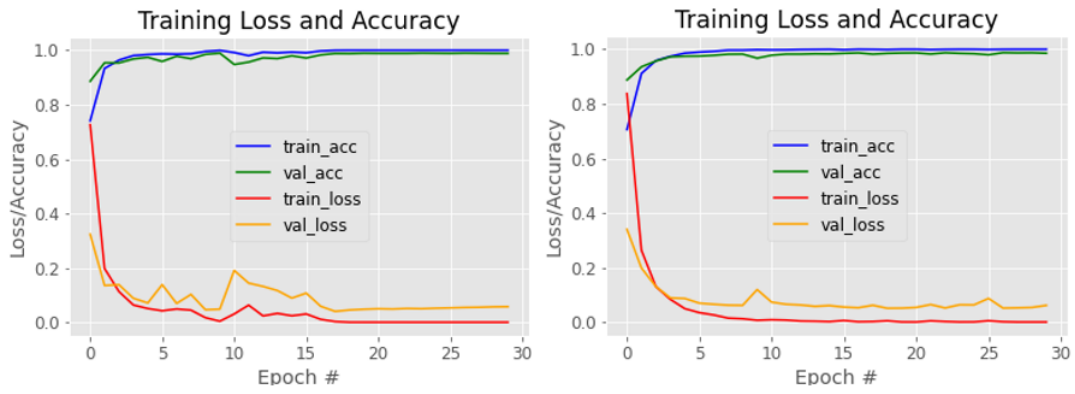


Figure B.1: VGG16 accuracy/loss using batch size 32 and learning rate $1e^{-4}$

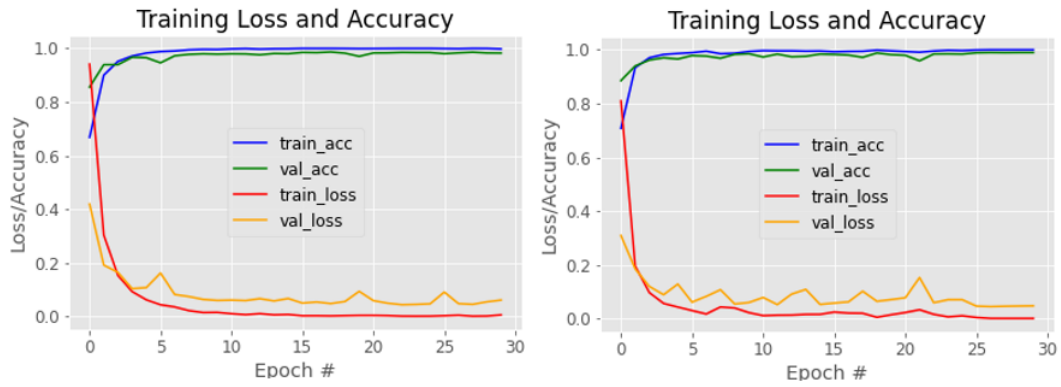


Figure B.2: VGG16 accuracy/loss using batch size 64 and learning rate $1e^{-4}$

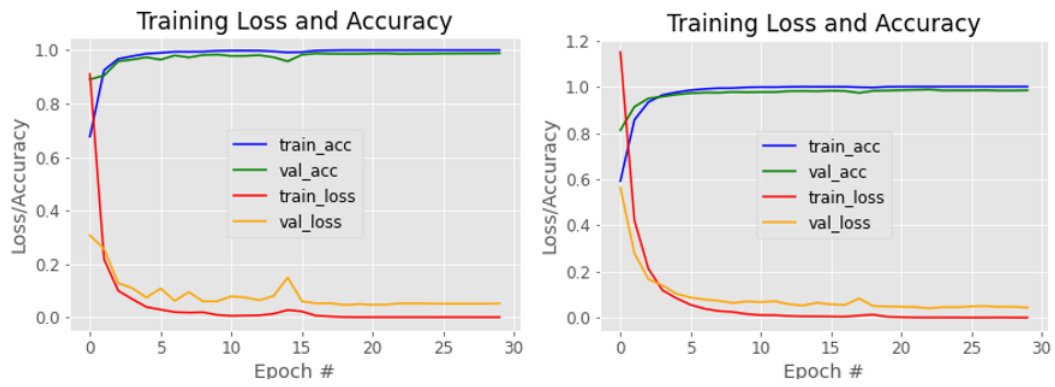


Figure B.3: VGG16 accuracy/loss using batch size 128 and learning rate $1e^{-4}$

B.2 Inception_V3

B.3 Learning Rate 0.0001

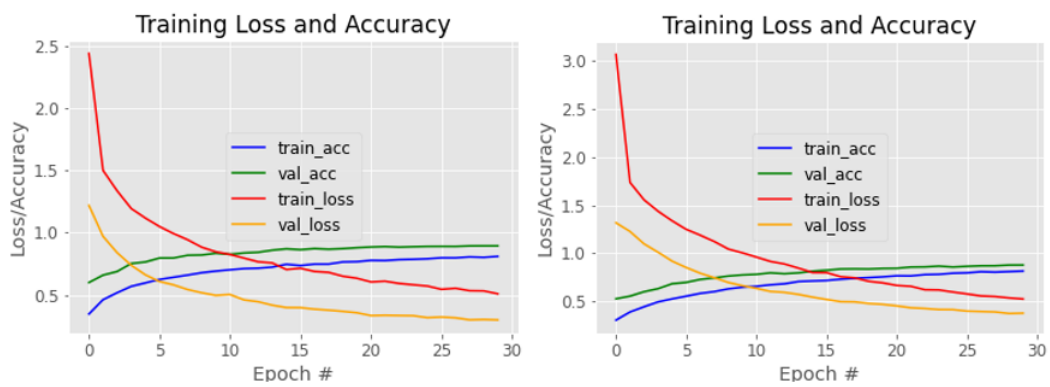


Figure B.4: Inception_V3 accuracy/loss using batch size 32 and learning rate $1e^{-4}$

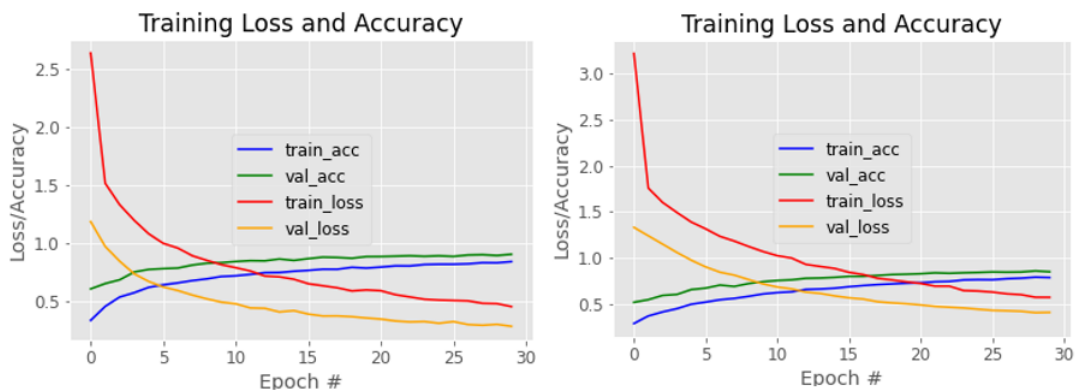


Figure B.5: Inception_V3 accuracy/loss using batch size 64 and learning rate $1e^{-4}$

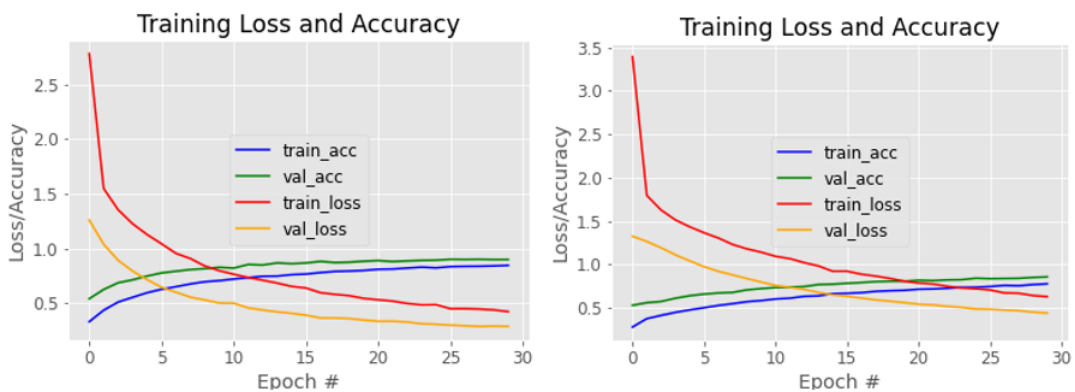


Figure B.6: Inception_V3 accuracy/loss using batch size 128 and learning rate $1e^{-4}$

Appendix C

Training and Testing Accuracy/loss Comparison Among Models

C.1 Experiment One

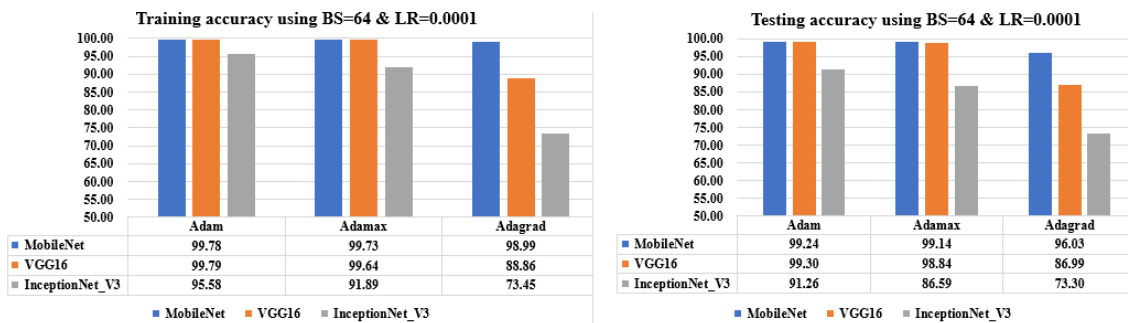


Figure C.1: Training and testing accuracy comparison using BS 64 and LR $1e^{-4}$

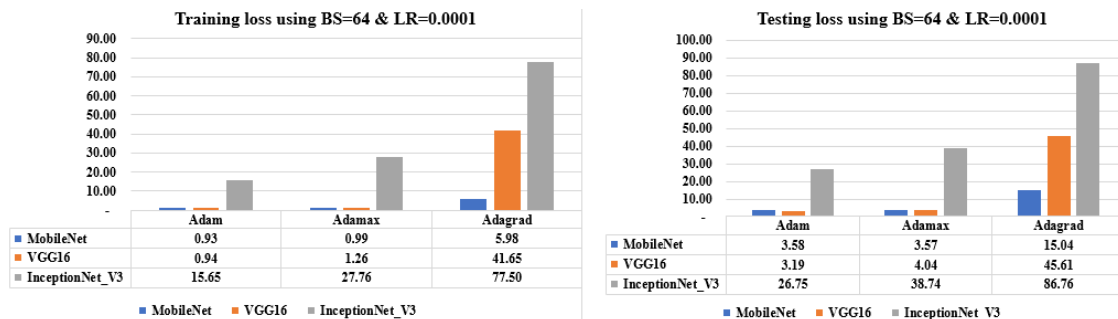


Figure C.2: Training and testing loss comparison using BS 64 and LR $1e^{-4}$

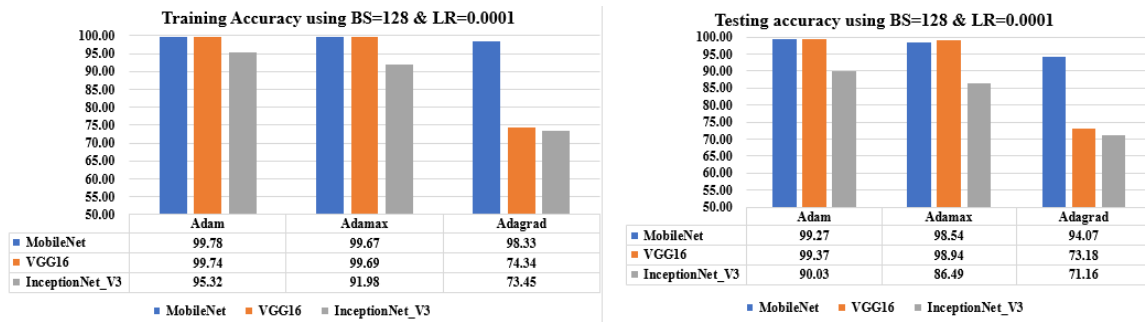


Figure C.3: Training and testing accuracy comparison using BS 128 and LR $1e^{-4}$

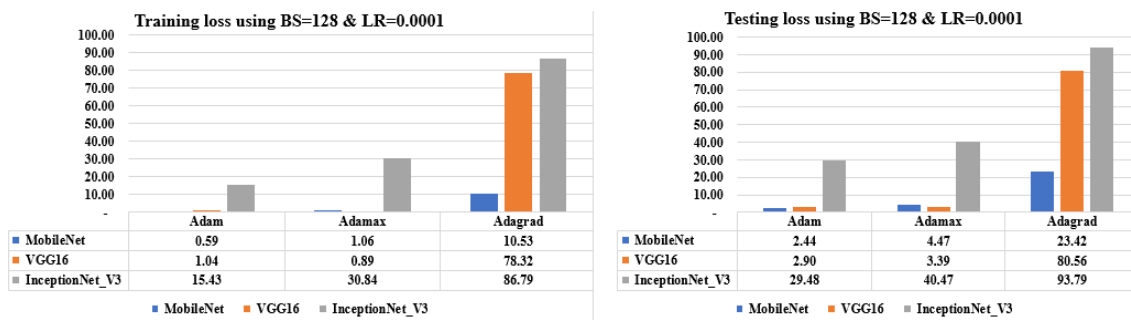


Figure C.4: Training and testing loss comparison using BS 128 and LR $1e^{-4}$

C.2 Experiemnt Two

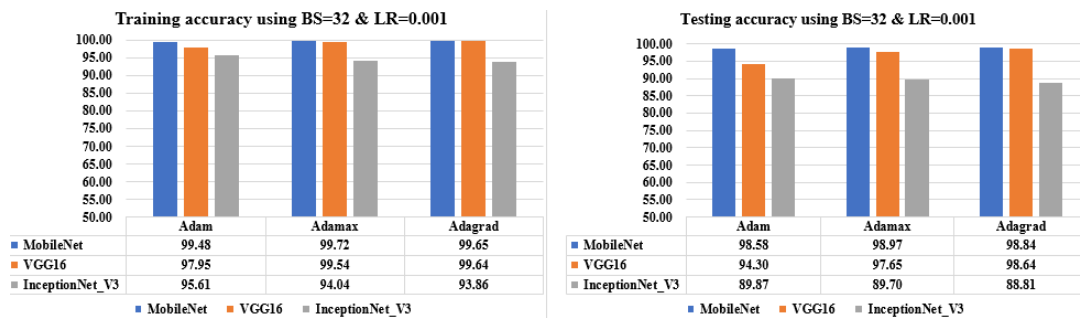


Figure C.5: Training and testing accuracy comparison using BS 32 and LR $1e^{-3}$

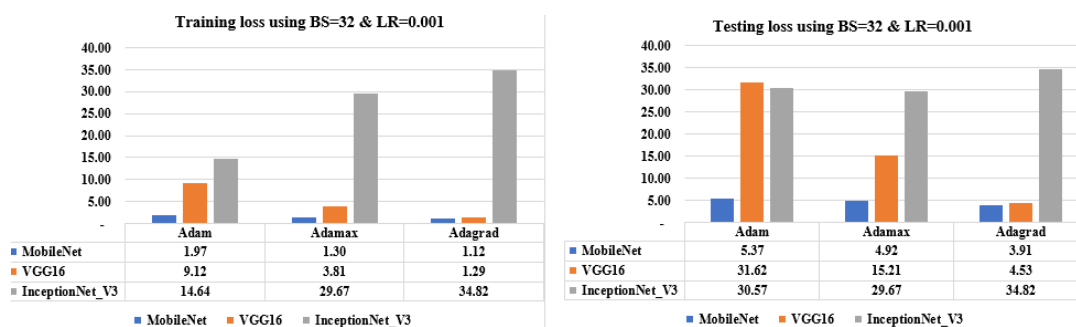


Figure C.6: Training and testing loss comparison using BS 32 and LR $1e^{-3}$

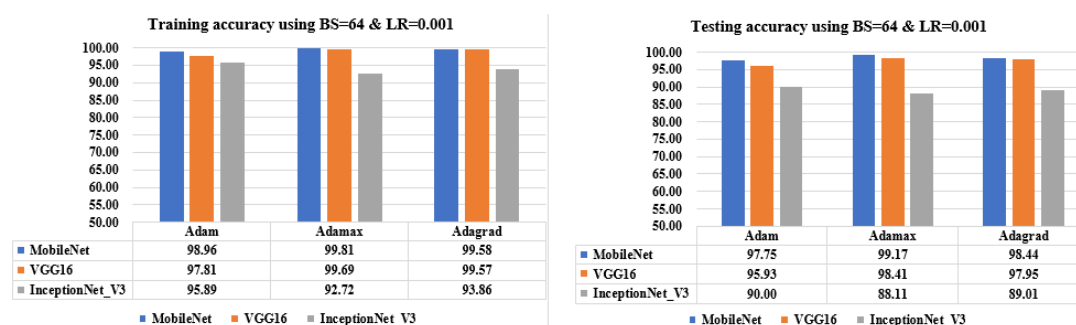


Figure C.7: Training and testing accuracy comparison using BS 64 and LR $1e^{-3}$

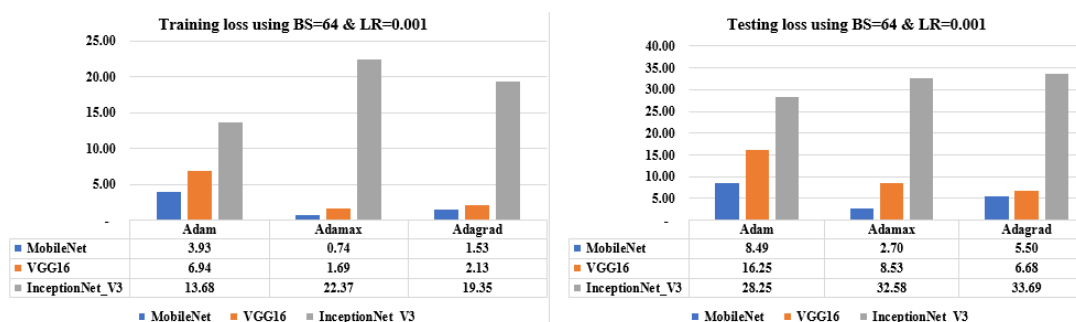


Figure C.8: Training and testing loss comparison using BS 64 and LR $1e^{-3}$

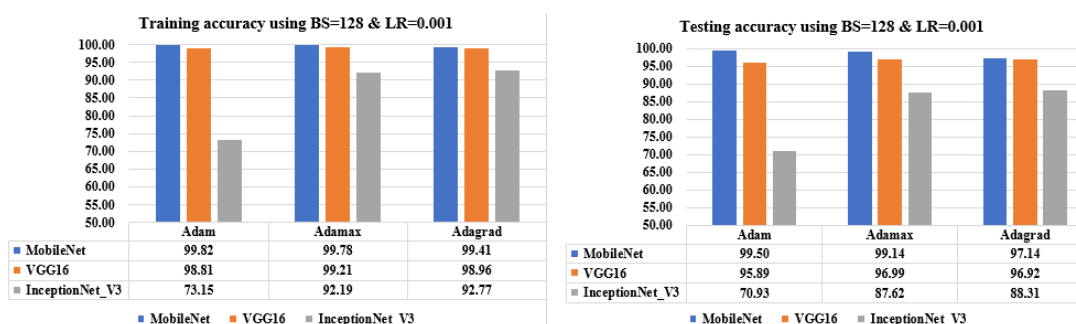


Figure C.9: Training and testing accuracy comparison using BS 128 and LR $1e^{-3}$

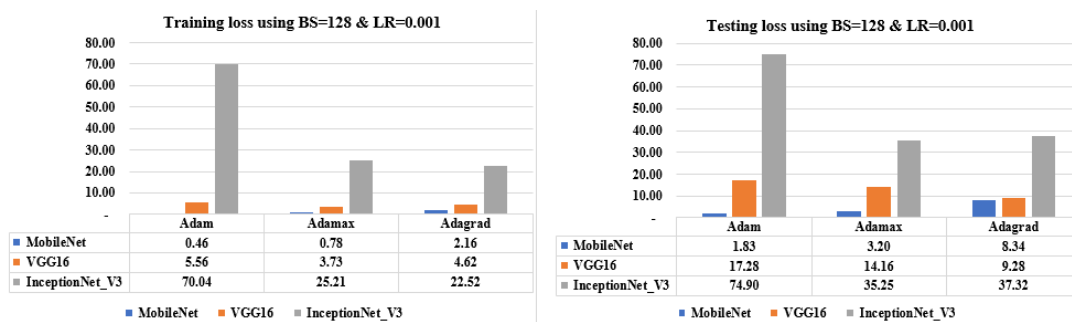


Figure C.10: Training and testing loss comparison using BS 128 and LR $1e^{-3}$

Appendix D

Model Evaluation

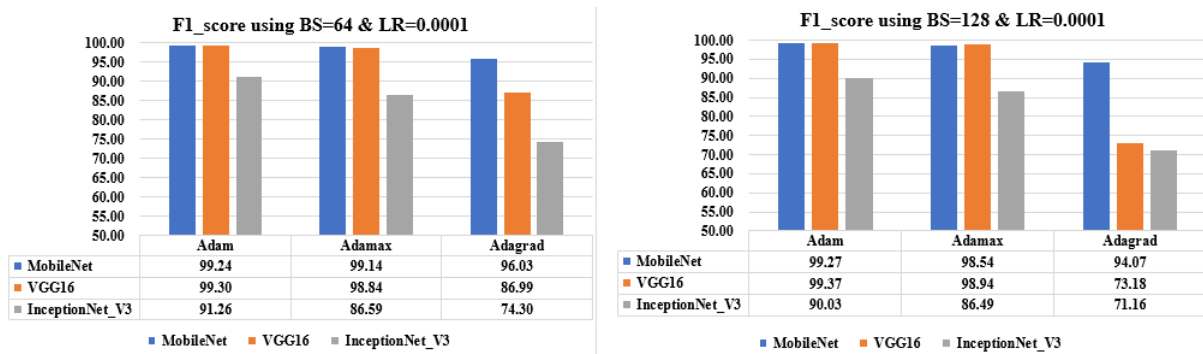


Figure D.1: F1 measure using BS 64, 128 and LR $1e^{-4}$

D.1 F1-Score, Precision and Recall using Learning Rate 0.0001

Table D.1: Mobile net F1-Score, Precision and Recall using LR $1e^{-4}$ with different BS

Batch size	Optimizers	Precision			Recall			F1_score		
		MA	MIA	WA	MA	MIA	WA	MA	MIA	WA
32	Adam	99.39	99.27	99.28	99.25	99.27	99.27	99.32	99.27	99.27
	Adamax	99.54	99.44	99.44	99.53	99.44	99.44	99.54	99.44	99.44
	Adagrad	98.33	98.01	98.02	97.96	98.01	98.01	98.14	98.01	98.01
64	Adam	99.33	99.24	99.24	99.40	99.24	99.24	99.37	99.24	99.24
	Adamax	99.29	99.14	99.14	99.13	99.14	99.14	99.20	99.14	99.14
	Adagrad	95.78	96.03	96.05	96.03	96.03	96.03	95.90	96.03	96.03
128	Adam	99.46	99.27	99.27	99.33	99.27	99.27	99.39	99.27	99.27
	Adamax	98.66	98.54	98.54	98.65	98.54	99.54	98.65	98.54	98.54
	Adagrad	93.33	94.07	94.17	94.65	94.07	94.07	93.94	94.07	94.08

Table D.2: VGG16 F1-Score, Precision and Recall using LR $1e^{-4}$ with different BS

Batch size	Optimizers	Precision			Recall			F1_score		
		MA	MIA	WA	MA	MIA	WA	MA	MIA	WA
32	Adam	99.07	99.07	99.07	99.17	99.07	99.07	99.12	99.07	99.07
	Adamax	98.98	99.01	99.02	99.12	99.01	99.01	99.04	99.01	99.01
	Adagrad	95.87	95.70	95.69	95.65	95.70	95.70	95.75	95.70	95.69
64	Adam	99.46	99.30	99.31	99.24	99.30	99.30	99.35	99.30	99.30
	Adamax	98.71	98.84	98.85	99.05	98.84	98.84	98.88	98.84	98.84
	Adagrad	88.38	86.99	87.14	84.57	86.99	86.99	86.13	86.99	86.79
128	Adam	99.60	99.37	99.37	99.45	99.37	99.37	99.52	99.37	99.37
	Adamax	99.02	98.94	98.94	99.13	94.94	98.94	99.07	98.94	98.94
	Adagrad	78.24	73.18	74.83	63.95	73.18	73.18	67.72	73.18	71.24

Table D.3: Inception_V3 F1-Score, Precision and Recall using LR $1e^{-4}$ with different BS

Batch size	Optimizers	Precision			Recall			F1_score		
		MA	MIA	WA	MA	MIA	WA	MA	MIA	WA
32	Adam	91.40	89.87	90.01	89.93	89.87	90.01	89.87	89.82	89.63
	Adamax	90.46	88.48	88.73	86.77	88.48	88.48	88.41	88.48	88.40
	Adagrad	81.12	66.82	73.55	54.11	66.82	66.82	60.02	60.82	64.43
64	Adam	92.37	91.26	91.37	90.36	91.26	91.26	91.30	91.26	91.25
	Adamax	89.86	86.59	87.17	93.88	86.59	86.59	86.51	86.59	86.52
	Adagrad	71.77	74.30	75.84	76.64	74.30	74.30	73.35	74.30	74.20
128	Adam	91.37	90.03	90.15	89.66	90.03	90.03	90.43	90.03	90.02
	Adamax	88.61	86.49	86.79	83.76	86.49	86.49	85.91	86.49	86.40
	Adagrad	69.45	71.16	73.58	73.68	71.16	71.16	70.30	71.16	71.08

D.2 F1-Score, Precision and Recall using Learning Rate 0.001

Table D.4: Mobile Net F1-Score, Precision and Recall using LR $1e^{-3}$ with different BS

Batch size	Optimizers	Precision			Recall			F1_score		
		MA	MIA	WA	MA	MIA	WA	MA	MIA	WA
32	Adam	98.95	98.58	98.58	98.64	98.58	98.58	98.79	98.58	98.58
	Adamax	99.22	98.97	98.98	99.12	98.97	98.97	99.16	98.97	98.97
	Adagrad	99.03	98.84	98.85	98.94	98.84	98.84	98.98	98.84	98.84
64	Adam	98.11	97.75	97.80	97.88	98.75	97.75	97.96	97.75	97.74
	Adamax	99.26	99.17	99.17	99.14	99.17	99.17	99.20	99.17	99.17
	Adagrad	98.76	98.44	98.44	98.44	98.44	98.44	98.60	98.44	98.44
128	Adam	99.61	99.50	99.50	99.58	99.50	99.50	99.59	99.50	99.50
	Adamax	99.14	99.14	99.14	99.10	99.14	99.14	99.12	99.14	99.14
	Adagrad	97.41	97.48	97.49	97.71	97.48	97.48	97.59	97.48	97.48

Table D.5: VGG16 F1-Score, Precision and Recall using LR $1e^{-3}$ with different BS

Batch size	Optimizers	Precision			Recall			F1_score		
		MA	MIA	WA	MA	MIA	WA	MA	MIA	WA
32	Adam	94.61	94.30	94.46	94.23	94.30	94.30	94.33	94.30	94.28
	Adamax	97.51	97.65	97.65	97.23	97.65	97.65	97.37	97.65	97.65
	Adagrad	98.74	98.64	98.64	98.53	98.64	98.64	98.63	98.64	98.64
64	Adam	99.14	98.94	98.95	98.88	98.94	98.94	99.01	98.94	98.94
	Adamax	98.61	98.41	98.41	98.36	98.41	98.41	98.48	98.41	98.41
	Adagrad	98.34	97.95	98.00	97.69	98.95	97.95	97.99	97.95	97.95
128	Adam	95.87	95.89	95.94	95.39	95.89	95.89	95.60	95.89	95.89
	Adamax	96.73	96.99	97.00	96.80	96.99	96.99	96.76	96.99	96.99
	Adagrad	97.53	96.92	96.97	96.55	96.92	96.92	97.01	96.92	96.92

 Table D.6: Inception_V3 F1-Score, Precision and Recall using LR $1e^{-3}$ with different BS

Batch size	Optimizers	Precision			Recall			F1_score		
		MA	MIA	WA	MA	MIA	WA	MA	MIA	WA
32	Adam	91.40	89.87	90.01	89.93	89.87	90.01	89.87	89.82	89.63
	Adamax	90.46	88.48	88.73	86.77	88.48	88.48	88.41	88.48	88.40
	Adagrad	81.12	66.82	73.55	54.11	66.82	66.82	60.02	6.82	64.43
64	Adam	92.37	91.26	91.37	90.36	91.26	91.26	91.30	91.26	91.25
	Adamax	89.86	86.59	87.17	93.88	86.59	86.59	86.51	86.59	86.52
	Adagrad	71.77	74.30	75.84	76.64	74.30	74.30	73.35	74.30	74.20
128	Adam	91.37	90.03	90.15	89.66	90.03	90.03	90.43	90.03	90.02
	Adamax	88.61	86.49	86.79	83.76	86.49	86.49	85.91	86.49	86.40
	Adagrad	69.45	71.16	73.58	73.68	71.16	71.16	70.30	71.16	71.08

D.3 Accuracy and Loss for Training and Testing Datasets

Table D.7: Mobile Net training and testing accuracy using LR $1e^{-4}$ with different BS

Batch size	Optimizers	Training accuracy	Testing accuracy	Training loss	Testing loss
32	Adam	99.78	99.27	1.09	3.29
	Adamax	99.84	99.44	0.86	2.76
	Adagrad	99.45	98.01	1.91	6.32
64	Adam	99.78	99.24	0.93	3.58
	Adamax	99.73	99.14	0.99	3.57
	Adagrad	98.99	96.03	5.98	15.04
128	Adam	99.78	99.27	0.59	2.44
	Adamax	99.67	98.54	1.06	4.47
	Adagrad	98.33	94.07	10.53	23.42

Table D.8: VGG16 training and testing accuracy using LR $1e^{-4}$ with different BS

Batch size	Optimizers	Training accuracy	Testing accuracy	Training loss	Testing loss
32	Adam	99.74	99.07	1.15	4.87
	Adamax	99.71	98.77	1.23	4.54
	Adagrad	94.26	91.92	22.56	26.89
64	Adam	99.79	99.30	0.94	3.19
	Adamax	99.64	98.84	1.26	4.04
	Adagrad	88.86	86.99	41.65	45.61
128	Adam	99.78	99.37	1.04	2.90
	Adamax	99.69	98.94	0.89	3.39
	Adagrad	74.34	73.18	78.32	80.36

Table D.9: Inception_V3 training and testing accuracy using LR $1e^{-4}$ with different BS

Batch size	Optimizers	Training accuracy	Testing accuracy	Training loss	Testing loss
32	Adam	96.12	90.53	15.49	28.39
	Adamax	94.16	88.87	18.33	33.73
	Adagrad	77.41	74.80	75.50	85.23
64	Adam	95.58	91.26	15.63	26.75
	Adamax	91.89	86.59	27.76	38.74
	Adagrad	76.48	74.30	77.50	86.76
128	Adam	95.32	90.03	15.43	29.48
	Adamax	91.98	86.49	30.84	40.47
	Adagrad	73.45	71.16	86.79	93.79

 Table D.10: Mobile Net training and testing accuracy using LR $1e^{-3}$ with different BS

Batch size	Optimizers	Training accuracy	Testing accuracy	Training loss	Testing loss
32	Adam	99.48	98.58	1.97	5.37
	Adamax	99.72	98.97	1.30	4.92
	Adagrad	99.65	98.84	1.12	3.91
64	Adam	98.96	97.75	3.93	8.49
	Adamax	99.81	99.17	0.74	2.70
	Adagrad	99.58	98.44	1.53	5.50
128	Adam	99.82	99.50	0.46	1.83
	Adamax	99.78	99.14	0.78	3.20
	Adagrad	99.41	97.48	2.16	8.34

Table D.11: VGG16 training and testing accuracy using LR $1e^{-3}$ with different BS

Batch size	Optimizers	Training accuracy	Testing accuracy	Training loss	Testing loss
32	Adam	97.95	94.30	9.12	31.62
	Adamax	99.54	97.65	3.81	15.21
	Adagrad	99.64	98.64	1.29	4.53
64	Adam	97.81	95.93	6.94	16.25
	Adamax	99.69	98.41	1.69	8.53
	Adagrad	99.57	97.95	2.13	6.68
128	Adam	98.81	95.89	5.56	17.28
	Adamax	99.21	96.99	3.73	14.16
	Adagrad	98.96	96.92	4.62	9.28

 Table D.12: Inception_V3 training and testing accuracy using LR $1e^{-3}$ with different BS

Batch size	Optimizers	Training accuracy	Testing accuracy	Training loss	Testing loss
32	Adam	95.61	89.87	14.64	30.57
	Adamax	94.07	89.70	19.15	29.67
	Adagrad	93.86	88.81	19.21	34.82
64	Adam	95.89	90.00	13.68	28.25
	Adamax	92.72	88.11	22.37	32.58
	Adagrad	93.86	89.01	19.35	33.69
128	Adam	73.15	70.93	70.04	74.90
	Adamax	92.19	87.62	25.21	35.25
	Adagrad	92.77	88.31	22.52	37.32

Selected Medicinal Plants

Table D.13: Selected medicinal plants

No.	Plant names	Part used
1	Absh	Seed
2	Adago	Unidentified
3	Ader	Leaf
4	Amera	Root, leaf
5	Anfar	Leaf
6	Aserkush	Root
7	Ashket	Leaf
8	Astenagr (maye zelefa)	Leaf, seed
9	Azohareg	Root, leaf
10	Besobila	Leaf, Seed
11	Bsana	Bark, leaf
12	Chqugn	Unidentified
13	Dama kesie	Leaf
14	Dedho	Bark
15	Dgita	Root, leaf
16	Dnblal	Seed
17	Enboy	Un identified
18	Endahula	Root, leaf
19	Enjori	Root
20	Enkoy	Bark, leaf
21	Enslal	Un identified

Table D.14: Selected medicinal plants continued

No.	Plant names	Part used
22	Eret	Leaf, Seed
23	Etse music	Seed
24	Etse zewie	Root
25	Feto	Seed
26	Gesho	Leaf
27	Gshta	Leaf
28	Grawa	Leaf
29	Gziewa	Leaf
30	Hareg eresa	Leaf
31	Lenquata	Bark
32	Lmch	Leaf
33	Msrch	Root, leaf
34	Nech bahr zaf	Leaf
35	Qentafa	Leaf
36	Qies bedeye	Leaf
37	Qnbo	Leaf
38	Qnchb	Root
39	Qulqual	Un identified
40	Rosmery	Leaf
41	Sama	Leaf
42	Smiza	Root, leaf
43	Telenj	Leaf
44	Tenadam	Leaf, seed
45	Tero	Un identified
46	Tql gomen	Leaf
47	Tqur azmud	Seed
48	Yayt hareg	Un identified
49	Yemdr enboy	Root
50	Yehabesha tsd	Seed
51	Yeset qest	Root
52	Zeytuna	Bark

Appendix E

Sample Fragmented Python Code

E.1 Pre-processing

```
import numpy
from numpy import loadtxt
from numpy import asarray
from numpy import save
train_image = []
with tf.device('/gpu:0'):
    for i in range(train.shape[0]):
        img = image.load_img('/content/drive/My Drive/colab/15000_data_with_25_plants/'+train['plants'][i]+' .jpg',
                             target_size=(128,128,3))
        img = image.img_to_array(img)
        img = img/255
        train_image.append(img)
X = np.array(train_image)
# save to npy file
save('15000_data_with_25_plants.npy', X)
```

Figure E.1: Pre-processing

E.2 Load image features and splitting target labels

```
from numpy import load
# load saved image feature
X = load('data.npy')
X.shape

y = np.array(train.drop(['plants', 'parts'],axis=1))
y.shape
```

Figure E.2: Load image features and splitting target labels

E.3 Classification Reports

```

from sklearn.metrics import multilabel_confusion_matrix
from sklearn.metrics import classification_report
print(multilabel_confusion_matrix(y_test_array, y_pred_array))
target_names = ['root', 'barck', 'leaf', 'seed', 'un_identied']
# print(classification_report(y_test_array, y_pred_array, labels=[0, 1, 2,3,4]))
print(classification_report(y_test_array, y_pred_array, target_names=target_names))

```

Figure E.3: Classification report

E.4 Micro, Macro, and Weighted Average

```

import numpy as np
y_gt = y_test_array
y_pr = y_pred_array
from sklearn.metrics import f1_score, precision_score, recall_score
p_macro = precision_score(y_gt, y_pr, average="macro")
p_micro = precision_score(y_gt, y_pr, average="micro")
p_weighted = precision_score(y_gt, y_pr, average="weighted")
r_macro = recall_score(y_gt, y_pr, average="macro")
r_micro = recall_score(y_gt, y_pr, average="micro")
r_weighted = recall_score(y_gt, y_pr, average="weighted")
f1_macro = f1_score(y_gt, y_pr, average="macro")
f1_micro = f1_score(y_gt, y_pr, average="micro")
f1_weighted = f1_score(y_gt, y_pr, average="weighted")
print('=====  
Precision Score =====')
print("P_macro: {:.4f}".format(p_macro))
print("P_micro: {:.4f}".format(p_micro))
print("P_weighted: {:.4f}".format(p_weighted))
print('=====  
Recall Score =====')
print("R_macro: {:.4f}".format(r_macro))
print("R_micro: {:.4f}".format(r_micro))
print("R_weighted: {:.4f}".format(r_weighted))
print('=====  
F1 Score =====')
print("F1_macro: {:.4f}".format(f1_macro))
print("F1_micro: {:.4f}".format(f1_micro))
print("F1_weighted: {:.4f}".format(f1_weighted))

```

Figure E.4: Evaluation metrics