African University of Science and Technology [AUST]

*Knowledge is Freedom*

# CASSAVA LEAF DISEASE CLASSIFICATION WITH DEEP LEARNING

**A Thesis Presented to the Department of Computer Science**

**African University of Science and Technology**

**In Partial Fulfilment of the Requirements for the Degree of**

**Master of Computer Science**

**By**

**EFFIONG BLESSING UDUAKOBONG**

**(40803)**

**Abuja, Nigeria**

**August 2021**

## CERTIFICATION

This is to certify that the thesis titled “Cassava Leaf disease classification with deep learning” submitted to the school of postgraduate studies, African University of Science and Technology (AUST) Abuja, Nigeria for the award of the master’s degree is a record of original research carried out by Effiong Blessing Uduakobong in the Department of Computer Science.

**African University of Science and Technology [AUST]**

***Knowledge is Freedom***

**APPROVAL BY**

**Supervisor**

Surname: Csató

First name: Lehel

Signature 

**The Head of Department**

Surname: Rajesh

First name: Prasad

Signature 

## ABSTRACT

Inspired by the current trend in artificial intelligence and neural network research, and with the increase in accuracy of models with deep neural network architecture (DNN), our work is dedicated to developing and training a deep neural network to extract meaningful patterns of diseases from leaves. We show how DNNs are applied to classification problems – classifying cassava leaf disease with a form of advanced convolutional neural networks with compound scaling (Efficient Nets) using supervised learning approach.

## ACKNOWLEDGEMENTS

I would like to express my gratitude to African University of Science and Technology for their devotion my studies, my supervisor, Professor Lehel Csató, for introducing me to the world of Artificial Intelligence and the Machine Learning universe, his endless enthusiasm, patience, understanding and guidance through independent research, which rendered this thesis successful. I would like especially thank Dr Rajesh Prasad, Head of the Department for his guidance. Also, to my partner Mr. Areji Jonathan Gebechukwu, for coaching me in my mathematical background and the entire colleagues of mine for the wise comments and encouragement or even constructive criticisms. To the DSN Africa team for their contributions towards expanding my knowledge in AI. Also, to all people I have met in AUST at the master level, for letting me learnt and endured their diversified ethnic and religious background I did not know. Thanks to my family and friends, for supporting me and being with me physically or otherwise. Thanks, a lot!

## DEDICATION

To Jonathan (Nkem), my entire family and friends who teach and have taught me in one way or the other.

Contents

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| AI | Artificial Intelligence |
| ANN | Artificial Neural Network |
| AUST | African University of Science and Technology |
| CNN | Convolutional Neural Network |
| DNN | Deep neural networks |
| HOD | Head of the Department |
| ML | Machine learning |
| NN | Neural network |
| RGB | Red, green, blue |
| SVM | Support Vector Machines |

# CHAPTER ONE

# INTRODUCTION

## 1.1 Background

Deep learning is inspired by the brain (Thorpe SJ), it does not attempt to copy be the brain but to understand the functionality of the brain and this inspiration is based on a conceptual level, as based on human understanding, in deep learning there are a lot of details about the brain whose relevance is unknown yet to human intelligence. The history of Artificial intelligence (A.I) and deep learning can be traced back to the history of using a network of binary neurons to perform logic in the early 40's, giving birth to the idea that the brain is a logical inference machine because the neurons are binary. From the rise of cybernetics and feedback mechanism in 1948, to perceptron in 1957, there have been differences, like the use of analog computers in contrast to the three-line python code we have now (Hadsell). The concept of back propagation did not exist until 1985 when they discovered the use of activation function that is continuous and differentiable to train multi-layer neural nets. (Y. ,. LeCun)

"*Machine learning as a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty, machine learning provides a deluge of data sets for automated methods of data analysis*" (Murphy),

This depicts that machine learning methods automatically detects patterns in data, can be used in cases of uncertainty for prediction. There are two main types of ML, supervised learning and unsupervised learning (Murphy). The other type of ML is called Reinforcement learning, deep learning is somewhat an aspect of supervised learning where a machine is trained by showing examples instead of programming, and close to 80-90% of deep learning models used supervised learning. Over the years there has been significant adaptations in this branch of deep learning, traversing from detection of object, to face and pedestrian detection (1993-2005), to training a robot to drive itself using convolutional neural networks (ConvNets) (Hadsell)

ConvNets (Hadsell) are networks whose layers are convolutional and perform simultaneous segmentation and recognition. It is a class of artificial neural networks (ANN), commonly used to analyze visual imagery. Its first successful application was mad by Yann LeCun in 1998 (Yann LeCun). The basic principle of deep learning is to get the raw data and turn it into something that is useful for the project you hope to achieve, deep learning does that by expanding the dimension of the representation of the data, so that the data is more likely to become linearly separable. General way to pre-process natural data in deep learning is by using processes like space tiling, polynomial classifiers, random projections, kernel machines.

" *A deep learning architecture trades space for time (breath for depth), it uses more sequential computation but less parallel computation*" (Leon Bottou).

Deep Learning architecture would rather achieve low computation time than have an increase in space usage, it solves a problem in less time by using more memory (storage space).

### 1.1.1 Python Programing Language

Python is a powerful programing language that is easy to learn, it became known to programmers since Guido van Rossum first released it in February 20 1991, it is a widely used interpreted, object-oriented, high-level programming language, whose design lay emphasis on code reusability. It is procedural, functional, structured and reflective, it is easy and intuitive open-source language that is suitable for any task. It is very good for scientific and numeric computing with SciPy and NumPy libraries, it also has an interactive shell (IPython) that features editing and recording of work sessions. Its latest version is python 3.95 and was released on May 3 2021. Python contains powerful libraries like Keras, Theano, and TensorFlow that are used for deep learning, it's simple syntax and readability promotes rapid testing of complex algorithms which is needed in machine learning.

Pytorch is a popular open-source deep learning framework, it is a python based scientific computing package, Pytorch can be used to write stochastic gradient, either from scratch or in a functional way or using the object-oriented way. Its advantages range from its automatic differentiation library that is very good for implementing neural networks to its replacements for NumPy to use

the power of GPU's as well as other accelerators. It is deeply integrated into python and is known for its fast, flexible and easy to learn path to production deployment. in order for one to fully understand the concept of Pytorch one has to first understand NumPy (a scientific library used for computing).

### 1.1.2 Classification

"*Classification is the problem of identifying to which of a set of categories (sub populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known*" (Tang).

The goal of classification is to learn to map a set of inputs to outputs, when the class is 2 then it is known as binary classification and if the class is greater than 2 then we would have a multi-class classification. When using classification in machine learning it is important to learn the boundary separating one class from the other classes.

### 1.1.3 Neural Networks and Deep Learning

Neural networks are a collection of neurons acting as processors. Convolutional neural networks are mainly used for image recognition as well as pattern recognition. The deeper the architecture of a Neural Network, the higher the number of hidden layers. Another concept of deep learning is parameterized functions, in most deep learning frameworks the parameter is usually implicit to the parameterized function in the object-Oriented versions of models. Deep learning also makes use of loss functions, which are things that we minimize during training. To understand more on deep learning and machine learning a pictorial difference is seen in figure 1.1 and figure 1.2.

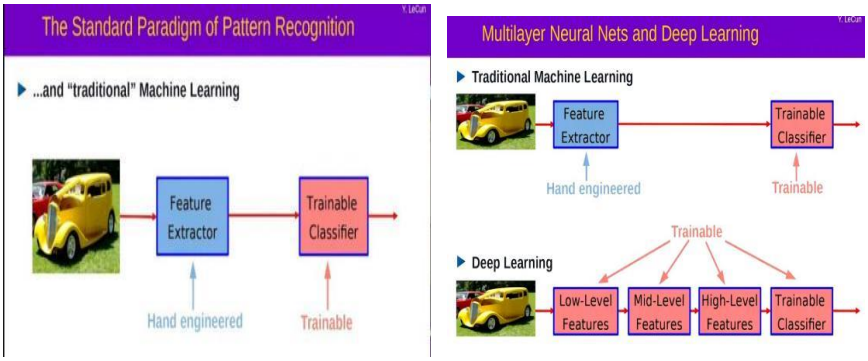


**Fig1.1 & Fig1.2 Traditional Machine VS Deep learning from** (Y. ,. LeCun)

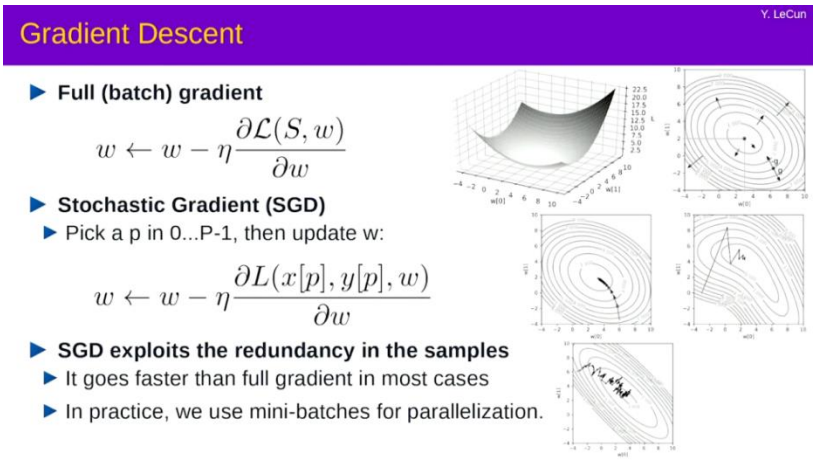


**Fig 1.3 Gradient Descent From** (Y. ,. LeCun)

Figure 1.3 displays the different gradient based method in machine learning. Gradient based method is a method that finds a minimum of a function assuming that you can easily compute the gradient of that function, assuming the function is more or less differentiable and must be continuous. Stochastic gradient descent is an optimization algorithm often used in machine learning applications to find the model parameters that correspond to the best fit between predicted and actual outputs. It's an inexact but powerful technique. In deep learning practice, we use stochastic gradient because it has some advantage over full gradient like

- It is easier to fit into memory with a single training sample being processed by the network

### 1.1.4 Convolutional Neural Networks with Efficient Nets

Efficient nets are used to achieve state of the art accuracy, they are a family of image classification models developed using AutoML and compound scaling. Efficient Net Pytorch is a Pytorch re-implementation of efficient net, that is consistent with the original TensorFlow implementation that enables the easy and flexible loading of weight from checkpoints.

This is a family of models that was scaled up from a new baseline network, that was designed from neural architectural search, it achieves a much better accuracy and efficiency than the previous ConvNets. According to (Quoc), efficient nets transfer well and achieve state of the art accuracy on CIFAR-100(91.7%), flowers (98.8%). Efficient Net shows that carefully balancing a network's depth, width and resolution leads to better performance. Using compound scaling coupled with automl to scale up CNN's in a structured manner creates a smaller and faster model.

### 1.1.5 Cassava Plant

Cassava is one of the most significant food security crops in sub-Saharan Africa, is also a staple food crop in Africa, Asia, and Latin America, it has proven to be one of the most dependable crops in a number of countries, it is produced either for raw materials for industries, as animal feeds, for exports as well as for human consumption. There are various diseases that the cassava plant is prone to, from cassava Blight, cassava bud necrosis, root rot disease, brown and white leaf spots, African cassava mosaic disease and so on.

## 1.2 Research Objective and Motivation

Image classification is a popular aspect of problem solving in many cases, over the years object and image classification as well as pattern recognition as been used in many fields. Recent studies have shown that viral food crop disease have the ability to halt crop production entirely, hence the need to successfully classify these diseases in the bid to solve crop disease crisis, or at least minimize and administer early treatment to the crop. Existing methods of disease detection, requires farmers to solicit the help of government-funded agricultural experts to visually inspect and diagnose the plants. This thesis focuses on classifying cassava plant diseases, using deep learning methods, to extract useful patterns from images of infected cassava crops, that identifies these infected crops from the healthy ones using multi-classification models, to help farmers quickly identify these diseases and save their crops.

One objective of this research (1) is to build a deep neural network using pytorch with python programming language, to train a deep learning model that uses images of cassava diseases to detect the kind of disease that is associated with the plant using already labelled classes of images of both health and infected cassava crops. Another objective (2) of this work is to extract meaningful patterns from the images that are as closely related to the disease pattern of the classes of the deep neural network model.

In realizing the goal of this thesis, answering the question, how can a deep learning algorithm learn to generalize learnt features from the pre-labelled training data? is important.

A neural network is a function that learns the expected output for a given input from training datasets by determining parameters (weights and bias) by itself.

### 1.3 Importance of Research

Cassava is a very important cash and food crop in sub-Saharan Africa, production of cassava crops massively can only be efficiently done if the farmers are able to detect these diseases on time and treat the plants. This research will help farmers to be able to easily detect these diseases and treat them on time. This work will enable farmers to automate disease detection for cassava, as well as improve the well-being of crops and aid in increasing crop productivity for farmers.

# CHAPTER TWO

# REVIEW OF RELATED LITERATURE

## 2.1 Introduction

This chapter is dedicated to an overview of the requisite knowledge of deep learning, image classification and its applications, as well as concepts and the mathematical techniques needed to develop the thesis and works related to this research. Our concentrations on this work will be based on supervised techniques known as classification.

## 2.2 Feature Extraction and Pattern Recognition

"*Feature extraction is most critical because the particular features made available for discrimination directly influence the efficacy of the classification or recognition task"* (Choras).

Feature extraction directly influences classification, with it being one of the major branches of ML, features from particular input data may be redundant or less important once it is, we can save the cost of extracting it. coherence vector, moments based, colour histograms and correlogram are used for the extraction of features in images, out of these methods colour moment is the best in terms of simplicity, compactness and robustness in its technique of extracting features. An ideal feature extractor should be able to disentangle explanatory factors of variation of what you are observing. Using facial recognition, each facial movement is an independent factor of variation, hence a representation that represents each of those factors of variations individually. This goes to show that learning each independent explanatory factors of variation in a dataset is very important for knowledge representation, and is one ultimate goal in representation learning, which has still not been done by anyone till date, In figure 2.1, it shows an Ideal feature extractor that is used in pattern recognition.

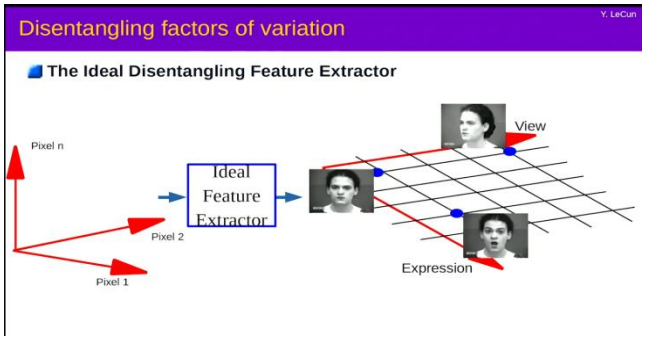


**Fig 2.1 Ideal feature extractor from** (Y. ,. LeCun)

The pre-processing stage is sometimes called feature extracting, and is usually done to ensure faster computing, instead of submitting large dataset for training, one can simply extract the important features, take into consideration that the new test data has to be pre-processed in the same steps as the training data, as well as take caution not to lose valid data when extracting feature from a test data else the overall accuracy of the system. Features are classified into two categories:

1. Local (low level) features, are usually small details of the image like point, curve or edges, etc.

2. Global features (high-level). High-Level features are built on top of local features to

detect objects and larger shapes in the image.

The problem of obtaining a pattern in a given image data is very important, inputs of images can be taken into a training model in form of vectors (x), recognizing patterns in images are done by using these input vectors to train in the training phase, and generate an output vector (y) which would be encoded in the same way as the target vector, once the model is trained it can then be used to determine the identity of new digit images (test set). Correctly categorizing new examples that differ from the already labelled examples used for training is known as generalization. To understand more on pattern recognition figure 2.2 displays components of a pattern recognition system.

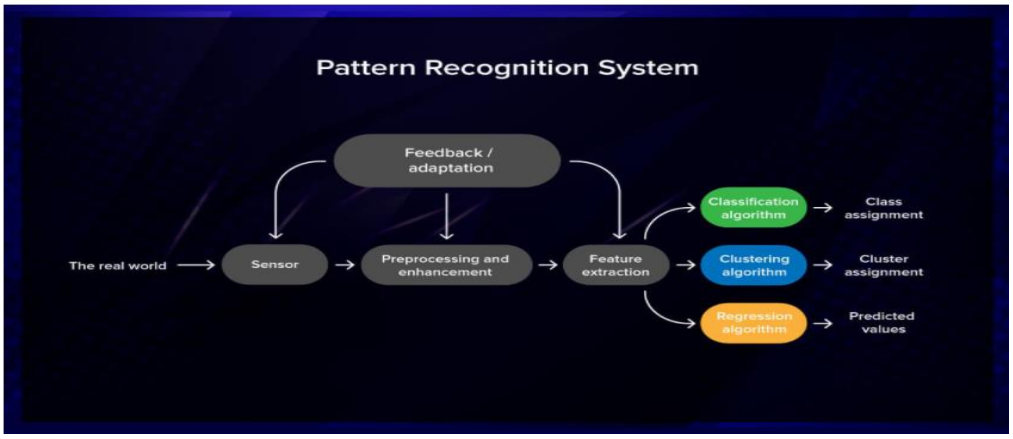


**Fig 2.2 Components of a pattern recognition system picture source: google images of pattern recognition**

Pattern recognition problems hopefully becomes more easier to solve when input variables are pre-processed to transform them into new spaces of variables, the central goal of pattern recognition is generalization (Bishop). When features are extracted, these features are then used as inputs for the pattern recognition algorithm. Unsupervised learning problems deals with training data that consists of a set of inputs vectors(x) without any corresponding target values (such as clustering problems, density estimation problems). Supervised learning deals with training data that consist of input vectors with their corresponding vector values (such as classification problems, regression problems). Finally, reinforcement learning is concerned with finding a suitable action to take in each situation to attain a reward, Pattern recognition is applied in these three types of learning. The field of pattern recognition is concerned with the automatic discovery of regularities in data using computer algorithms and with the use of these regularities to take actions such

In the 1970's and 1980's pattern recognition gained popularity, and it focused on how to make computer programs perform intelligent human tasks, at that time, it was the most innovative form of signal processing, with concepts like heuristics models, decision tree and quadratic discriminatory analysis. When working with learning domain, we use feature extraction and pattern recognition to make life easier. One important characteristics of the large dataset used is that they require a lot of computing resources to process them due to the large number of variables, so feature extraction helps to get the best features from these big data sets by selecting and combining variables into features, thereby effectively reducing the data, it helps in reducing redundant data from the dataset in order to build models with less machine's effort and increase the speed of

learning and generalization steps in machine learning. For Image processing we use feature extraction and pattern recognition algorithms to detect features on images to process them. We can say that pattern recognition is the beginning of intelligent programs

## 2.2 Machine Learning

In the early 90's many realized that there was a better and effective way of representing pattern recognition algorithms using probability and statistics. Machine learning uses mathematics, statistics and data as well as domain specific knowledge to solve complex problems. It is all about turning data into numbers and finding a pattern in these numbers using algorithms, the basic task of a machine learning is to create a model, that can classify or predict different patterns from data, these models are created by algorithms that run on data and are used to pattern predict. There are two types of machine learning, supervised learning and unsupervised learning. A machine learning model is immune to cognitive bias and fatigue, as one of its primarily goals is to make conclusions on a collection of data with the most minimal human intervention. Figure 2.3 displays the Classical Machine Learning structure.

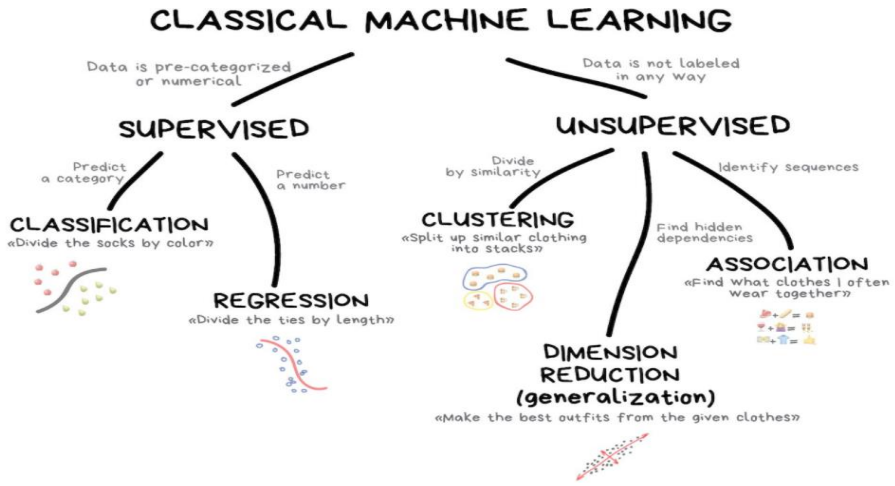


**Fig 2.3 Classical Machine Learning picture source: Dr Natalia Konstatinova's blog[1]**

[1] Alibaba Cloud Blog
( https:// www.alibabacloud.com/blog/deep-learning-vs-machine-learning-vs-pattern-recognition_207110 )
Accessed on July 5-2021


**Fig 2.4 Typical machine learning process picture source: Dr Natalia Konstatinova's Blog [2]**

In figure 2.4 showing typical machine learning process, A machine learning trains a collection of data, validates the data collection, and finally uses the probabilistic model derived using pattern recognition algorithms to test the data collection, a training set is used to train the model, a validation set is used to fine-tune the parameters of the model, and a testing data is used to check the accuracy of the system.

### 2.3 Deep Learning

In recent years, deep leaning has opened a new area of research for it has proved its outstanding performance in ML and pattern recognition. (Deng) defines deep learning as

"A *class of ML techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification"*. (Deng).

Deep learning is positioned in the intersections among the research areas of NNs, AI, graphical modelling, optimization, pattern recognition, and signal processing. Deep Learning was introduced with the objective of moving ML towards to one of its original goals, AI. Deep learning evolved from the acquisition of big data, the power of parallel and distributed computing, and sophisticated algorithms has in multitude ways facilitated major advances in domains such as image recognition, speech recognition, and natural language processing where the AI community struggled for many years (Y. B. LeCun). Figure 2.5 shows Deep-learning: general methods of machine learning

[2] Alibaba Cloud Blog
( https:// www.alibabacloud.com/blog/deep-learning-vs-machine-learning-vs-pattern-recognition_207110 )
Accessed on July 5-2021

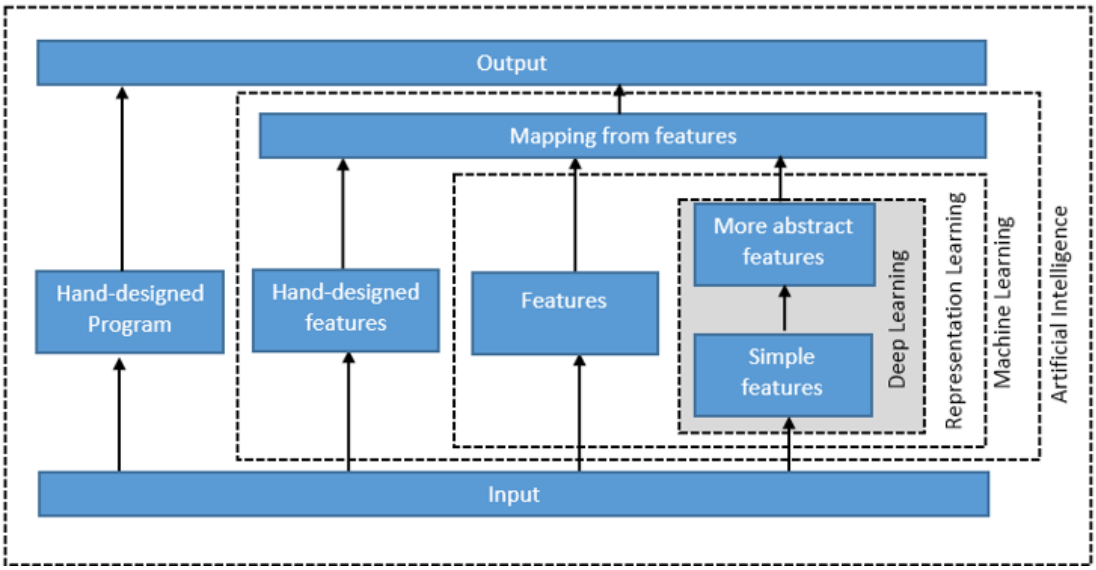


**Fig 2.5 Deep-learning: general methods of machine learning**

In deep learning, the idea is to learn feature levels of increasing abstraction with minimum human contribution (Bengio). When there is a lot of labelled data as well as a lot of computational power then a deep neural network can be trained using supervised learning but initializing the weights using unsupervised learning rather than random initialization, is much faster and gives a better result with fewer labelled data. The main idea behind deep learning is that it requires less manual interference and is intuitive (Alypaydin).

Deep neural networks use raw inputs, where each hidden layer combines the value of the preceding layer to learn more complicated function of the input. It is a specialized form of ML that uses artificial neural networks to process complex image, audio, and natural language. In deep learning, the idea is to learn feature levels of increasing abstraction with minimum human contribution (Bengio). A deep neural network is typically trained one layer at a time, The aim of each layer is to extract the salient features in the data that is fed to it, and a method such as the autoencoder is used, from the raw input it trains the autoencoder, then the encoded representation that was learned in its hidden layer is then used as inputs for training the next autoencoder, and this goes on till the final layer trained in a supervised manner with labelled data, after that they are assembled and the entire system is fine-tuned using the labelled data (Alypaydin). In figure 2.6, the pictorial difference between machine learning and deep learning is seen.

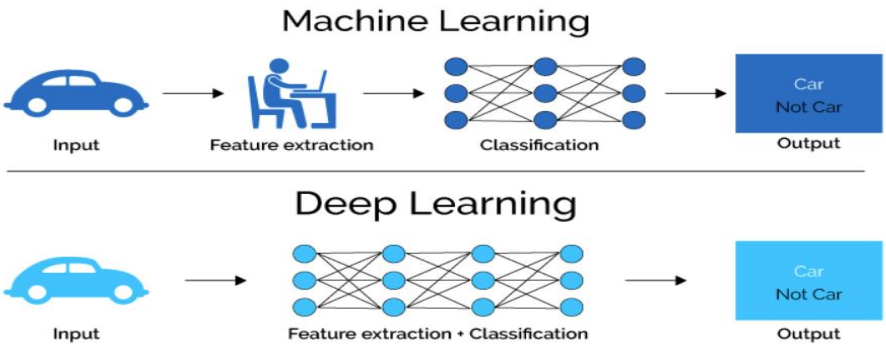


**Fig 2.6 Deep learning Vs Machine Learning picture source: levity Blog[3]**

Some refer to deep learning as a framework to unite the world, it requires minimal human intervention and bias as its parameters are solely based on statistics. It is only possible wit big data and an ample amount of computing power as well as strong arithmetic capabilities (GPU) to optimize the mode. In deep learning, concepts like neural pattern recognition are used, where artificial neural networks are used to learn complex nonlinear input output relations and adapt themselves to data.

### 2.3.1 Convolutional neural network (CNN)

Convolutional neural network (CNN) models have been used in large scale graphic recognition over recent years, it has the potential to improve plant disease phenotyping where the standard approach is visual diagnostics requiring specialized training. They are powerful image processing AI that uses deep learning to is used to perform generative and descriptive task using machine vision for both image and voice recognition. Its neural network is made after the operations of neurons in the human brain, their neurons are arranged like that of the frontal lobe (area responsible for processing visual stimuli in the human brain), this is done to avoid the problem of piecemeal image processing that is typically faced by the traditional neural network. Multilayer perceptron system is like the system used by CNN, it has 5 layers:

1. Input layer
2. Convo layer (Convo + RELU)
3. Pooling Layer Fully connected (Fc) layer

[3] Levity Blog (https://levity.ai/blog/difference-machine-learning-deep-learning)
Accessed July 06-2021

4. Softmax/logistic layer
5. Output layer

CNN removes the limitations of traditional neural networks, and increases efficiency for image processing, hence it makes a system that is more effective and simpler to train in image processing. Its grid like topology was specifically designed to process pixel data, it is used to enable sight to the computer, a CNN can be trained to do image analysis tasks, including scene classification, object detection and segmentation, and image processing. In order to understand how CNNs work, we'll cover three key concepts proposed by (Alzubaidi) :

- Local receptive fields: The region referred to as local receptive fields are a small region of input layer neurons connected to neurons in the hidden layer, it is translated across an image to create a feature map from the input layer to the hidden layer neurons.

- Shared weights and biases: the weights and bias values are the same for all hidden neurons in a given layer. This means that all hidden neurons are detecting the same feature, such as an edge or a blob, in different regions of the image.

- Activation and pooling: The activation step applies a transformation to the output of each neuron by using activation functions. Rectified linear unit (ReLU), is an example of a commonly used activation function. It takes the output of a neuron and maps it to the highest positive value or if the output is negative, the function maps it to zero. You can further transform the output of the activation step by applying a pooling step. Pooling reduces the dimensionality of the featured map by condensing the output of small regions of neurons into a single output. This helps simplify the following layers and reduces the number of parameters that the model needs to learn in the convnet framework shown in figure 2.7.

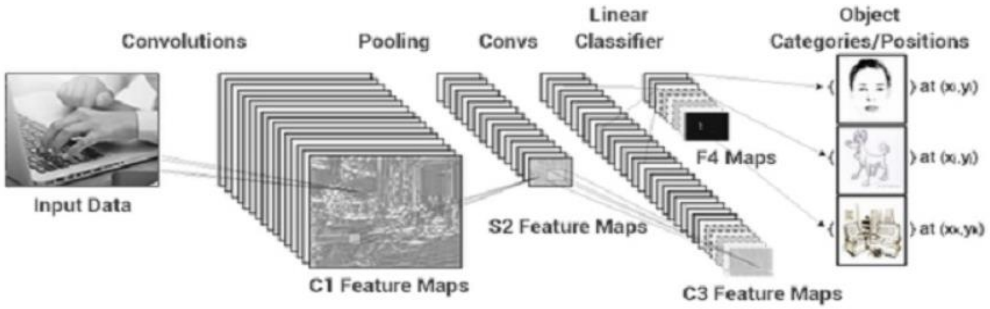


**Fig 2.7 ConvNet framework from Google picture source: google images of Convolution**

There are three ways CNN use to train convolutional network for image analysis:

- Training the model from scratch;
- Using transfer learning (based on the idea that you can use knowledge of one type of problem to solve a similar problem);
- Using a pretrained CNN to extract features for training a machine learning model.

It is essential for model assessment to be conducted in real world conditions if such models are to be reliably integrated with computer vision products for plant disease phenotyping in a CNN object detection model to identify foliar symptoms of diseases in cassava.

### 2.3.2 Terminologies In CNN

In as much as CNN's are part of neural networks, there are some important concepts that are peculiar to CNN, these are concepts that make them different other neural network, some of the main ones are explained below:

1. **CONVOLUTION KERNELS:** The kernel is an integral part of the layered architecture it is an operator applied to an entire image such that the information in the image encoded in pixels is transformed into real life scenario. The kernels are convolved together with input volumes to form activations and can be seen in figure 2.8.

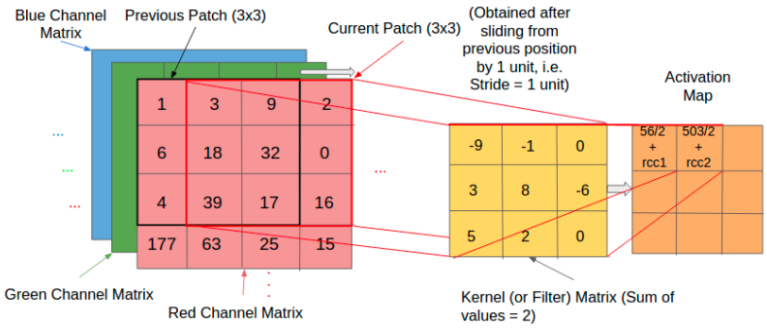


**Fig 2.8 Convolutional process with kernel picture source: Towards Data Science blog[4]**

2. **INPUT/OUTPUT VOLUMES;** An RGB image of say, 32 x 32 (width x height) pixels would have three matrices associated with each image, one for each of the colour channels. Thus, the image in its entirety, constitutes a three-dimensional structure called input volume. Conventionally, all images are seen as matrices with pixels values (with height and width), adding the different attributes with depths of separate colour channels of the RGB (3 colours), it forms the input into 3 dimensions one for each of the colour channel, this implies that the number of colours present in an image determines the dimensionality of an input volume. An RGB image with the dimensions four-units width and four-units height can be represented pictorially in figure 2.9 as:

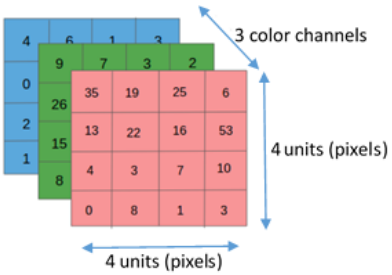


**Fig 2.9 RGB with 4 units picture source: Towards Data Science Blog[5]**

3. **EDGE DETECTION:** Every image has both vertical and horizontal edges which combines to form images, convolutional operation is used with some filters for detecting edges,

[4] Towards Data Science Blog (https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529) Accessed July 08-2021

[5] Towards Data Science Blog (https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529) Accessed July 08-2021

The convolution operation can be visualized in the following way. in figure 2.10 we see the Visualization of convolution.

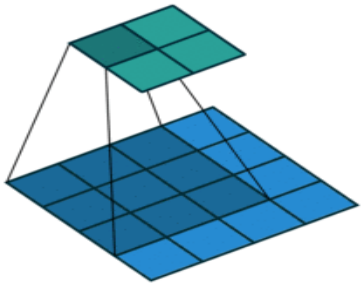

**Fig 2.10 Visualization of convolution picture source: Towards Data Science Blog[6]**

4. **RECEPTIVE FIELD:** This is a two-dimensional region that is specified in units, that extends to the depth of the entire input unit. They are used for network layer cross-section

operations and production of activation maps. this is seen in figure 2.11 showing the Conceptual Representation of receptive field**s.**

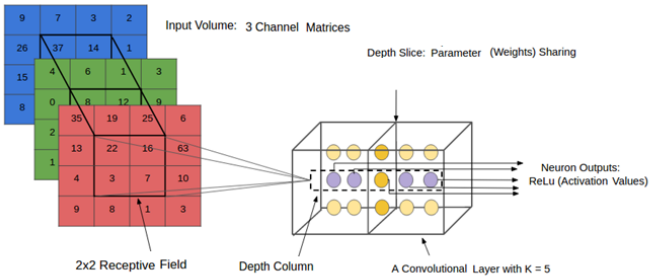


**Fig 2.11 Conceptual Representation of receptive fields picture source: Towards Data Science Blog[7]**

[6] Towards Data Science Blog (https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529) Accessed July 08-2021

[7] Towards Data Science Blog (https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529) Accessed July 08-2021

5. **PADDING AND STRIDE;** A stride is used to denote how many steps is taken in each step-in convolution; in default it is one. Padding is used for maintaining the dimension of output as in input, it is a process of adding zeros to the input symmetrically (zero padding), figure 2.12 displays Convolution with stride and figure 2.13 displays Convolution with stride and padding

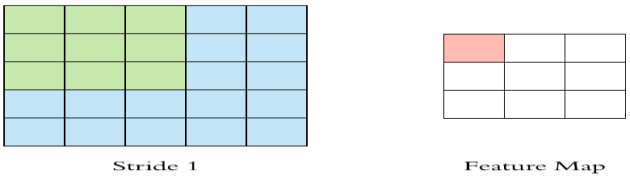


**Fig 2.12 Convolution with stride picture source: Towards Data Science Blog[8]**

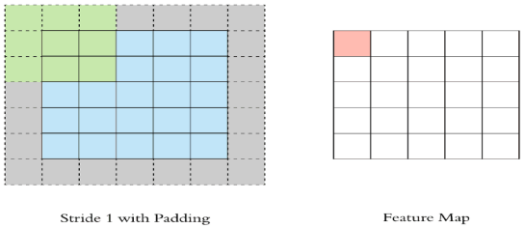


**Fig 2.13 Convolution with stride and padding picture source: Towards Data Science Blog[9]**

Padding is used so that the original size of the image is retained (avoid image shrinkage) and to ensure that corner features of the image or on the edges are not used in the output. While stride is the number of pixel shift over the input matrix.

[8]Towards Data Science Blog (https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529) Accessed July 09-2021

[9] Towards Data Science Blog (https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529) Accessed July 08-2021

**6.POOLING;** Pooling is performed in neural network to reduce variance and computational complexity, there are 3 basic pooling methods that are widely used.

- **Max pooling:** This selects the maximum pixel value of the batch
- **Average pooling:** This selects the average value of all the pixels in the batch
- **Min pooling:** This selects the minimum pixel value of the batch

The choice of pooling operation is dependent on the data at hand. Max pooling selects the brighter pixels from the image, by eliminating non-maximal values to reduce computation for the upper layer, Min pooling gives a better result for images with white background and black objects.

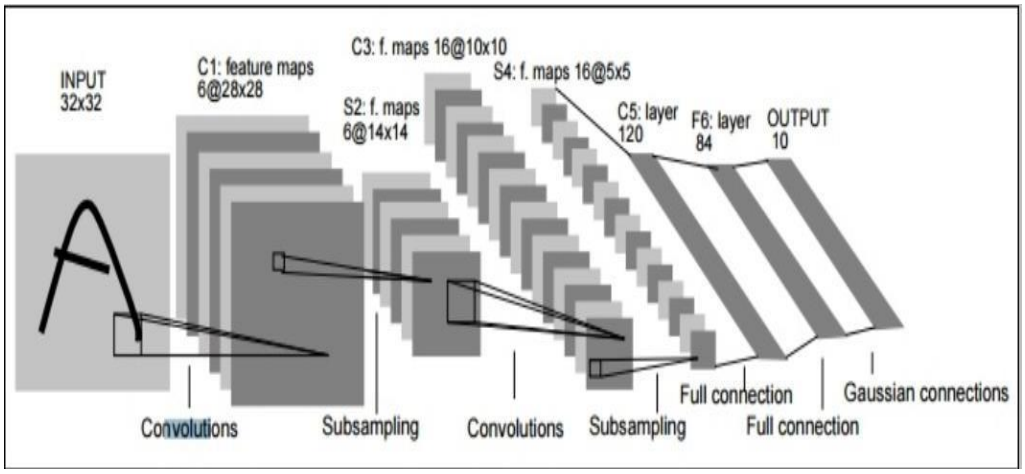


**Fig 2.14 LeNets-5 Architecture from Google**

In terms of its architecture figure 2.14 displays the LeNets-5 Architecture, it was the first well known convolution architecture ever published, from leNet-5, to the current convolutional neural nets with efficientnets that uniformly scales all dimensions and depth using compound coefficient, CNN's architecture has evolved over the years to create faster and more human like systems for use in all sectors. In as much as we say deep learning is directly inspired by the human brain, its

implementation is somewhat different as the brain are built using cells and neural networks are built using mathematical operators.

# CHAPTER THREE

# RELATED WORK ON APPLICATION OF DEEP LEARNING

Plant disease is an important factor in determining the yield of plants, and as it is well known, the area of agriculture is a very important field to man, with it being the centre and core of food production worldwide. It is due to the impact of the agriculture, that Artificial Intelligence (AI) tries to find more ways to make life easier for man. In recent years, deep learning has made breakthroughs in the area of digital image processing, and plant disease detection is a very important research content in machine vision, it basically entails using some machine vision equipment to acquire images, to check if there are diseases present in the images present in the dataset. A lot of machine learning models have been employed for the detection and classification of plant disease, but using deep learning architecture with several visualization methods to classify and detect plant diseases, has shown more accuracy than other branches of machine learning, and this has been done for different agricultural applications.

The problem of protecting plant disease efficiently is closely related to the problem of agricultural and climate change (K. A. Garrett). Plant disease tends to be concept of human experience rather than a purely mathematical definition, it is a kind of natural disaster that affects the growth of plants and causes plant death in most cases. the possibility of misidentifying and misclassifying plant disease with a high error rate, as well as the labour-intensive nature to identify and categorize diseases by the artificial eyes, gave rise to the proposed method of using deep learning-based methods to identify and classify plant disease. In recent years, there have been rapid increase in leaf diseases affecting crops, coupled with unpredictable climate and weather changes, this has led to low cultivation leaving the farmers to face loss. About 40% of worldwide crops are lost to pest and diseases annually according to food and agricultural organizations of the united-nations, crops are liable to encounter bacterial, fungal and viral infections but the rate of disease spread is dependent on the state of the crop and its vulnerability to infection. When crops are infected, they show some symptoms (rot, pale coloration, spots on leaf, blights, scabs, yellow coloration, etc.),

we use deep learning methods to identify and classify leaf diseases early to prevent further spread and termination of crops eventually.

In the detection of plant disease there are some questions that are important to answer in order to successfully detect, (What, Where, How) and these 3 questions are mutually inclusive and can be converted (Liu);

- What: This has to do with which disease we are trying to detect and on what plant, it corresponds to the classification task (describe the image globally through feature extraction)
- Where: This has to do with the part of the plant that we are focused on, whether the fruit, root, leaf, stem, it corresponds to the location task, is more concerned with the detection (detection stage that focuses on local description and structure learning)
- How: this corresponds to the segmentation of the data in computer vision

Many deep learning architectures has been developed over time for increased accuracy and efficiency and to bring about precision in agriculture, with noticeable increase in parameter size as well as accuracy below is a table showing the evolution of deep learning from its first architecture LeNet.

**TABLE 3.1 TABLE OF DEEP LEARNING EVOLUTION**

| **Deep Models** | **Learning Parameters** | **Key Features and Pros/Cons** |
|---|---|---|
| LeNet | 60k | First CNN model. Few parameters as compared to other CNNmodels. Limited capability of computation. |
| AlexNet | 60M | Known as the first modern CNN. Best image recognition performance at its time. Used ReLU to achieve better performance. Dropout technique was used to avoid overfitting. |
| OverFeat | 145M | First model used for detection, localization, and classification of objects through a single CNN. Large number of parameters as compared to AlexNet. |
| ZFNet | 42.6M | Reduced weights (as compared to AlexNet) by considering 7 × 7 kernels and improved accuracy. |
| VGG | 133M–144M | 3 × 3 receptive fields were considered to include more number of non-linearity functions which made decision function |

| Deep Models | Learning Parameters | Key Features and Pros/Cons |
|---|---|---|
| | | discriminative. Computationally expensive model due to large number of parameters. |
| GoogLeNet | 7M | Fewer number of parameters as compared to AlexNet model. Better accuracy at its time. |
| ResNet | 25.5M | Vanishing gradient problem was addressed. Better accuracy than VGG and GoogLeNet models. |
| DenseNet | 7.1M | Dense connections between the layers. Reduced number of parameters with better accuracy |
| EfficientNet | 1.6B | EfficientNet-B0,achieves 77.3% accuracy on ImageNet with only 5.3M parameters and 0.39B FLOPS. (Resnet-50 provides 76% accuracy with 26M parameters and 4.1B FLOPS)., and the proposed compound coefficient produce models that provide more accuracy. |
| SqueezeNet | 1.25M | Similar accuracy as AlexNet with 50 times lesser parameters. Considered 1 × 1 filters instead of 3 × 3 filters. Input channels were decreased. Large activation maps of convolution layers |
| Xception | 22.8M | A depth-wise separable convolution approach. Performed better than VGG, ResNet, and Inception-v3 models |
| MobileNet | 4.2M | Considered the depth-wise separable convolution concept. Reduced parameters significantly. Achieved accuracy near to VGG and GoogLeNet. |
| Modified/Reduced MobileNet | 0.5/0.54M | Lesser number of parameters as compared to MobileNet. Similar accuracy as compared to MobileNet. |
| VGG-Inception | 132M | A cascaded version of VGG and inception module. The number of parameters were reduced by substituting 5 × 5 convolution layers with two 3 × 3 layers. Testing accuracy was increased as compared to many well-known DL models like AlexNet, GoogLeNet, Inception-v3, ResNet, and VGG-16. |

(Saleem)

As seen on the table above, EfficientNet has more learning parameters than most, and in terms of performance, it shows more percentage of accuracy.

**TABLE 3.2 TABLE OF THEORITICAL COMPARISM OF EFFICIENTNET AND OTHER MODELS**

| Model | Top-1 Acc. | Top-5 Acc. | #Params | Ratio-to-EfficientNet | #FLOPS | Ratio-to-EfficientNet |
|---|---|---|---|---|---|---|
| **EfficientNet-B3** | **81.7%** | **95.6%** | **12M** | **1x** | **1.8B** | **1x** |
| ResNeXt-101 (Xie et al., 2017) | 80.9% | 95.6% | 84M | 7.0x | 32B | 18x |
| PolyNet (Zhang et al., 2017) | 81.3% | 95.8% | 92M | 7.7x | 35B | 19x |
| **EfficientNet-B4** | **83.0%** | **96.3%** | **19M** | **1x** | **4.2B** | **1x** |
| SENet (Hu et al., 2018) | 82.7% | 96.2% | 146M | 7.7x | 42B | 10x |
| NASNet-A (Zoph et al., 2018) | 82.7% | 96.2% | 89M | 4.7x | 24B | 5.7x |
| AmoebaNet-A (Real et al., 2019) | 82.8% | 96.1% | 87M | 4.6x | 23B | 5.5x |
| PNASNet (Liu et al., 2018) | 82.9% | 96.2% | 86M | 4.5x | 23B | 6.0x |
| **EfficientNet-B5** | **83.7%** | **96.7%** | **30M** | **1x** | **9.9B** | **1x** |
| AmoebaNet-C (Cubuk et al., 2019) | 83.5% | 96.5% | 155M | 5.2x | 41B | 4.1x |
| **EfficientNet-B6** | **84.2%** | **96.8%** | **43M** | **1x** | **19B** | **1x** |
| **EfficientNet-B7** | **84.4%** | **97.1%** | **66M** | **1x** | **37B** | **1x** |
| GPipe (Huang et al., 2018) | 84.3% | 97.0% | 557M | 8.4x | - | - |

Comparison of EfficientNet with existing networks for ImageNet Challenge. Source: (Shahid)

**TABLE 3.3 TABLE OF THEORITICAL COMPARISM OF EFFICIENTNET UPSCALED**

| Model | FLOPS | Top-1 Acc. |
|---|---|---|
| Baseline MobileNetV1 (Howard et al., 2017) | 0.6B | 70.6% |
| Scale MobileNetV1 by width ($w$=2) | 2.2B | 74.2% |
| Scale MobileNetV1 by resolution ($r$=2) | 2.2B | 72.7% |
| **compound scale ($d$=1.4, $w$=1.2, $r$=1.3)** | **2.3B** | **75.6%** |
| Baseline MobileNetV2 (Sandler et al., 2018) | 0.3B | 72.0% |
| Scale MobileNetV2 by depth ($d$=4) | 1.2B | 76.8% |
| Scale MobileNetV2 by width ($w$=2) | 1.1B | 76.4% |
| Scale MobileNetV2 by resolution ($r$=2) | 1.2B | 74.8% |
| **MobileNetV2 compound scale** | **1.3B** | **77.4%** |
| Baseline ResNet-50 (He et al., 2016) | 4.1B | 76.0% |
| Scale ResNet-50 by depth ($d$=4) | 16.2B | 78.1% |
| Scale ResNet-50 by width ($w$=2) | 14.7B | 77.7% |
| Scale ResNet-50 by resolution ($r$=2) | 16.4B | 77.5% |
| **ResNet-50 compound scale** | **16.7B** | **78.8%** |

Scaling up mobileNets and ResNet using compound scaling: Source: (Shahid)

### 3.2 Deep Learning Models

Traditional machine learning methods have been applied in plant disease prediction, in order to improve accuracy and yield better diagnostic results faster. From the year 2012 till date, different deep learning architectures have been proposed and implemented, with these models displaying an increase in larger training sets, having been evaluated using several performance metrics

#### 3.2.1 Deep Learning Implementation:

Deep-learning models can be implemented with or without visualization techniques, here are some works that were done without visualization techniques, Classification of maize leaf disease using CNN (Sibiya M.) and a histogram technique was used to show the significance of the model, also in the case of tomato leaf disease identification using deep learning, basic CNN architectures like ReSNet, AlexNet, and GoogleNet were used, and validation accuracy as well as training accuracy was plotted to show the system's performance. inception-v3 (a new deep Learning model) was used for detection of cassava disease (Ramcharan A.). LeNet architecture was used for detecting banana leaf diseases and classification accuracy was used to evaluate the model (Alsayed). Figure 3.1 displays the evolution of deep learning models since 2012.

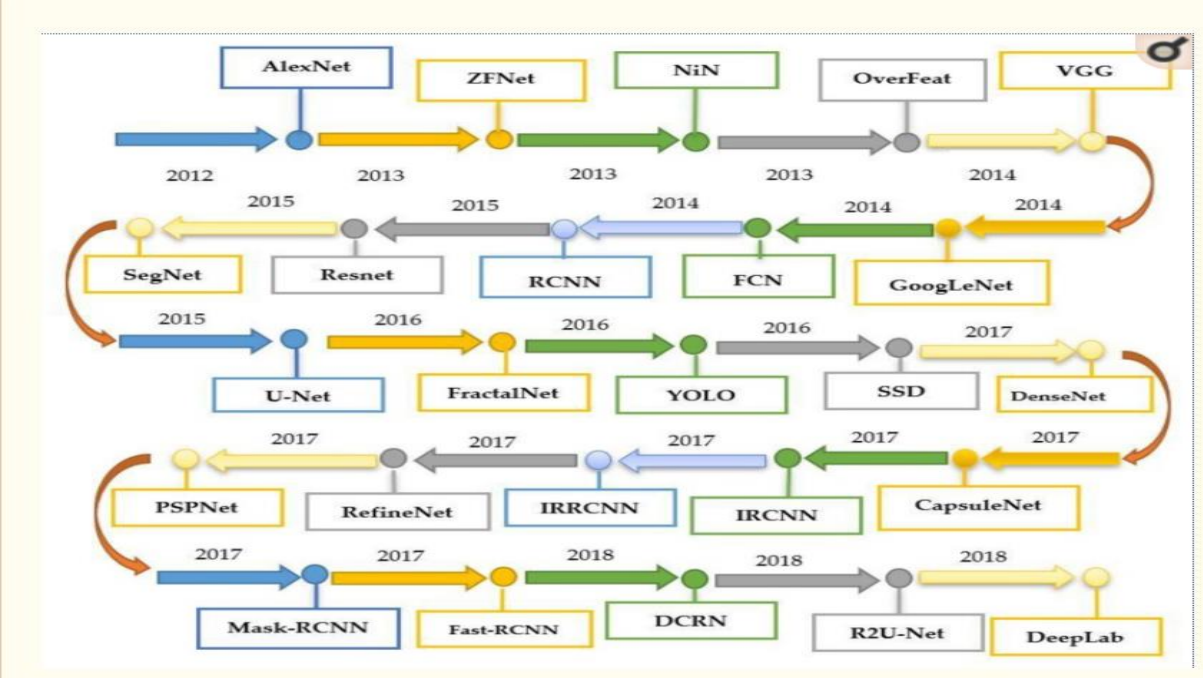


**Fig 3.1 Evolution Of Deep Learning Models since 2012 picture source:** (Saleem)

Several research directions have been carried out on deep learning models using visualization techniques for a clearer understanding of plants' diseases like a modified LeNet model was used to detect olive plant diseases. The segmentation and edges maps were used to spot the diseases in the plants by (Albert C.Cruz). A new DL model was introduced in (Brahimi M.) named teacher/student network and proposed a novel visualization method to identify the spots of plant diseases. In (Alsayed), a banana leaf disease and pest detection were performed by using three CNN models (ResNet-50, Inception-V2 and MobileNet-V1) with Faster-RCNN and SSD detectors. According to (DeChant C.), different combinations of CNN were used and presented heat maps as input to the diseased plants' images and provided the probability related to the occurrence of a particular type of disease, ROC curve was used to evaluate the performance of the model and feature maps for rice disease were also included in the paper. Below are some

visualization techniques that have been used.

**TABLE 3.4 TABLE OF VISUALIZATION TECHNIQUES/ MAPPINGS**

| **Visualization Techniques/Mappings** | | **References** |
|---|---|---|
| Classification and localization of diseases by bounding boxes | | (Fuentes A.) |
| Heat maps were used to identify the spots of the disease | | (DeChant C.) |
| Feature map for the diseased rice plant | | (Lu Y.) |
| Feature map for spotting the diseases | | (Ghosal S.) |
| Image segmentation method | | (Ma J.) |
| Segmentation map and edge map | | (Albert C.Cruz) |

In the above table, bounding boxes, feature maps, heat maps, edge maps and segmentation maps are the list of visualization techniques and mappings that can be used with deep learning models.

Classification involves two phases (training and prediction) Within the training phase, data is used and analysed into a set of features based on the feature generation models such as the vector space model for textual data and fine-tuned to create a predictive model, while during the prediction set untrained data is channelled to the model for prediction. Classification of plant disease using images involves a 4-step process according to (Murphy). the first one being to take images (RBG) after which colour transformation structure is developed, then masking and removal of green pixels are performed using threshold value in step two, step three has to do with creating useful segments using segmentation process and texture statistics and finally, step four is applying classifier algorithms to the extracted features to classify the diseases.

Another way of detecting and classifying plant diseases is by using histogram matching based on edge detection and colour feature. In 2019, (Adebayo. Segun) worked on cassava disease detection with machine learning, training ML model to detect cassava disease, to predict if the leaf is healthy or not, with a coarse gaussian support vector machine with an accuracy of 61.6%, and it was trained to detect two diseases cassava mosaic disease (CMD), and cassava bacteria blight disease (CBBD). Our work is on 5 classes of diseases not just two.

In summary, we have presented different approaches centred on deep learning, deep learning models and how much they have evolved and in what way, as well as classifications using deep learning models and applications in plant diseases based on various published papers and books.

# CHAPTER FOUR

# RESEARCH METHODOLOGY

## 4.1 Deep Learning Toolkits / Libraries and Architecture

Convolutional Neural Network (CNN) is one of the most popular architecture for deep learning for images, these are several criteria for determining the best toolkit for deep learning. Toolkits are tools that will let you play with the data, train your models, discover new methods, as well as create your own algorithms, with the continuous evolution of ML and deep learning, there have been several toolkits and frameworks created over the past years, like these below with some being more used and popular than others ; TensorFlow (python), Theano (python), Caffe, Deep learning4j (Java with Scala), Digits, Keras, MXNET, Lasagna(python), Scikit-Learn, CNTK (Cognitive Network Toolkit), Chainer, Pylearn (python), Torch (C)Pytorch (Python). In this thesis work, we used Pytorch.

### 4.1.1 Pytorch Packages

Pytorch is an open-source deep learning library, written in python, it was developed by Facebook, it is the python font-end to the torch computational engine, whose machine learning library was developed in Lua. It shares some similarities with TensorFlow in terms of it being a low-level API focused on deep learning. It has robust debugging experience and is structured in a way that is native to python (Erickson), It easily makes use of data pre/post processing and visualization tools from Torch, below are some of the advantages of Pytorch package:

- Minimum dependency: python backend ready to run, easy for fast prototyping.
- It allows users to step through models in a way that Keras and TensorFlow don't permits.
- deep nets training.
- It has higher developer productivity.

- Has data parallelism, can support computational work on multiple CPU's and GPU cores
- Correctness: all computation layers are unit-tested.
- Provides high performance of Torch with good GPU support with a friendlier python front end.
- Has dynamic computational graph support.
- Provides hybrid front ends (graph mode and eager mode)
- It has deep integration to keep points that can allow more flexibility in how networks are constructed.

PyTorch has native ONNX support and can export models in the standard Open Neural Network Exchange format, it also has Cloud support.

### 4.1.2 Useful Pytorch Libraries

Communities supporting development in computer vision, reinforcement learning, and much more have built various libraries for Pytorch to bolster it's reach as a fully-featured deep learning library for both research and production purposes (Dhiraj k). Here are a few examples of popular libraries:

Gpytorch: is a highly efficient and modular implementation with GPU acceleration. It's implemented in Pytorch and combines Gaussian processes with deep neural networks.

AllenNLP: is an open-source NLP research library, built on Pytorch.

## 4.2 Dataset Compilation

Our proposed system consists of five different phases, Image Acquisition, Image Pre-processing, Image segmentation, Feature Extraction and Feature Classification. (Mariyappan.)

The above stages have already been carried out on the cassava leaf disease dataset by Makerere AI lab in Kampala with data collected being 21,367 images,

### 4.2.1 Collection of Data

Crop leaves generally gets affected when it encounters a disease and it is considered as a first sign that disease may spread to rest of the crops. Most images were crowd sourced from farmers taking photos of their farms, and annotated by experts at the National crops resource research institute in

collaboration with AI lab at Makerere University, Kampala. In figure 4.1 , this shows a data image from the lab.

**Figure 4.1 Test training image from Makerere AI lab**

In this work we would be analysing and classifying the following diseases as the dataset we are provided with, to create and optimize our network with contains 21,367 observations of cassava leaf images and their corresponding disease classifications (already labelled). The five categories that the leaf images fall under are:

Cassava Bacterial Blight (CBB), label (0).
Cassava Brown Streak Disease (CBSD), label (1).
Cassava Green Mottle (CGM), label (2).
Cassava Mosaic Disease (CMD), label (3).
Healthy cassava, label (4).

This can be seen in figure 4.2 below, as it shows the different labels of diseases of cassava in our dataset.

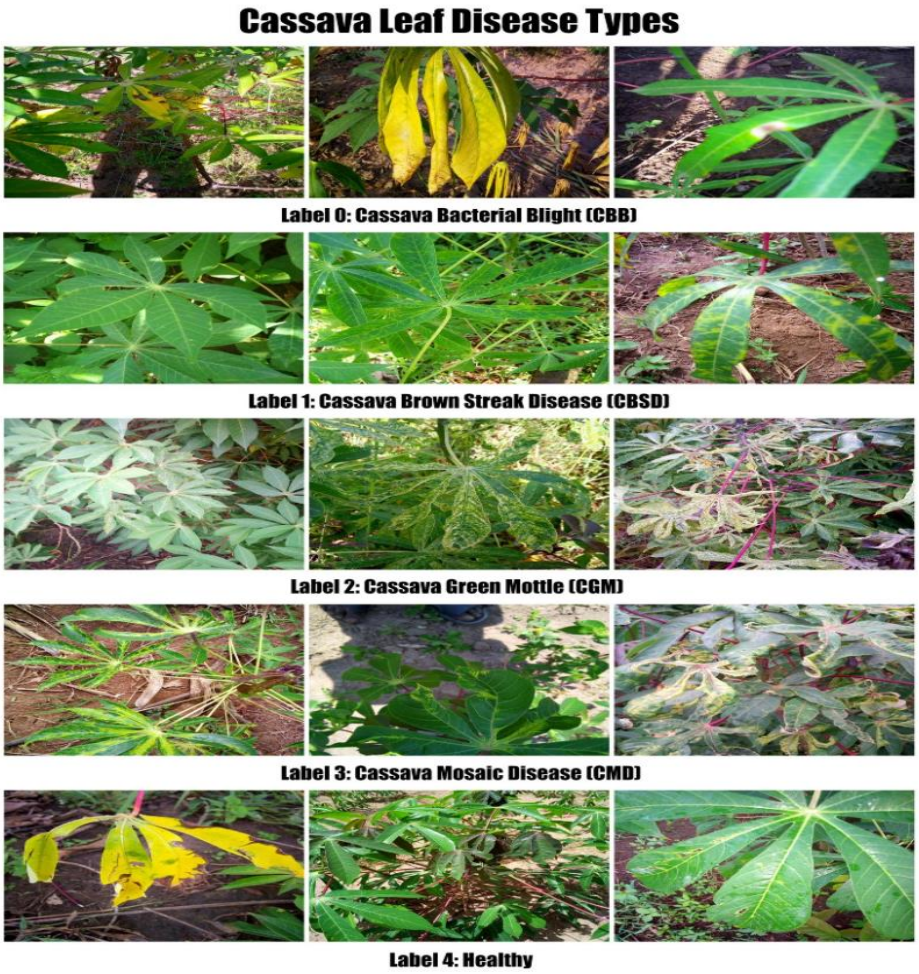


**Figure 4.2 Cassava Leaf Images for different classes in the dataset**

### 4.2.2 Data Preparation

Data was split into two folders test folder and train folders, when we train a deep learning model, we don't just want it to learn to model the training data. We want it to generalize to data it hasn't seen before, there's a very convenient way to measure an algorithm's generalization performance: we measure its performance on a held-out test set, consisting of examples it hasn't seen before, the 80-20 training-testing split was made on our dataset, as we are creating a generative model, that learn more from training with 80% of the data, and testing our model with 20% of our data that it has not seen before.

Data preparations has three steps:

#### 4.2.2.1 Data Duplication and Bias Detection

In this step, data duplication is a process that eliminates excessive copies of data, and this has already been done on our work by the AI lab in Makerere. data bias was noticed removed from our dataset. The following figure shows the labelled images with their amount of images in our dataset: classes 0, 1, 2, and 3 had altogether less, than 9000 items, while class 33 had more, than 13000 data items alone.

While most of the classes such as CBSD, CGM, and Healthy have between 2100 and 2600 images for each of them, there are over 13000 images of CMD stricken cassava plants which accounts for nearly 61.5% of our entire set of training data, it is reasonable to assume the more likely scenario that CMD is by far the most common condition, and should allow us to create a very naive baseline model that simply classifies all images as being afflicted with it.

#### 4.2.2.1 Data Discrimination (Done by the Makerere AI lab)

The segregations were made based on the following features for importance for the training

- Colour
- Texture
- Shape /size

Using EfficientNets with compound scaling, this allows us to load a pre-trained set for our model. The figure 4.3 below shows the data representation of the dataset in the system.

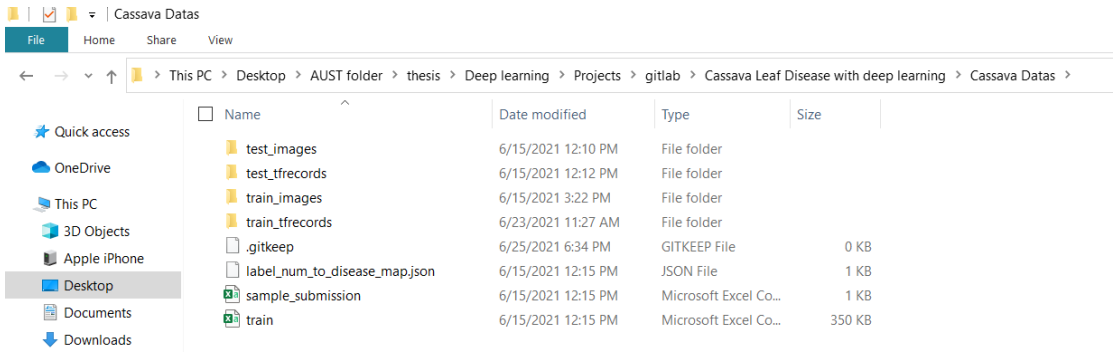


**Figure 4.3 Data Representation**

### 4.2.3 Instruments Used

1. Kaggle's cassava leaf disease dataset

2. Jupyter notebook

3. Gitlab

### 4.3 Defining Network Architecture

In this proposed system, we propose deep CNN (EfficientNets). we explore the structures and the building blocks of convolutional neural networks.

### 4.3.1 Neural Architecture Design

The image below shows the degree of classification accuracy (CA) when efficientnet is used as to when other deep learning models. Table 4.1 Neural Architecture Table

| Model | Top-1 Acc. | Top-5 Acc. | #Params | Ratio-to-EfficientNet | #FLOPS | Ratio-to-EfficientNet |
|---|---|---|---|---|---|---|
| **EfficientNet-B0** | **77.3%** | **93.5%** | **5.3M** | **1x** | **0.39B** | **1x** |
| ResNet-50 (He et al., 2016) | 76.0% | 93.0% | 26M | 4.9x | 4.1B | 11x |
| DenseNet-169 (Huang et al., 2017) | 76.2% | 93.2% | 14M | 2.6x | 3.5B | 8.9x |
| **EfficientNet-B1** | **79.2%** | **94.5%** | **7.8M** | **1x** | **0.70B** | **1x** |
| ResNet-152 (He et al., 2016) | 77.8% | 93.8% | 60M | 7.6x | 11B | 16x |
| DenseNet-264 (Huang et al., 2017) | 77.9% | 93.9% | 34M | 4.3x | 6.0B | 8.6x |
| Inception-v3 (Szegedy et al., 2016) | 78.8% | 94.4% | 24M | 3.0x | 5.7B | 8.1x |
| Xception (Chollet, 2017) | 79.0% | 94.5% | 23M | 3.0x | 8.4B | 12x |
| **EfficientNet-B2** | **80.3%** | **95.0%** | **9.2M** | **1x** | **1.0B** | **1x** |
| Inception-v4 (Szegedy et al., 2017) | 80.0% | 95.0% | 48M | 5.2x | 13B | 13x |
| Inception-resnet-v2 (Szegedy et al., 2017) | 80.1% | 95.1% | 56M | 6.1x | 13B | 13x |
| **EfficientNet-B3** | **81.7%** | **95.6%** | **12M** | **1x** | **1.8B** | **1x** |
| ResNeXt-101 (Xie et al., 2017) | 80.9% | 95.6% | 84M | 7.0x | 32B | 18x |
| PolyNet (Zhang et al., 2017) | 81.3% | 95.8% | 92M | 7.7x | 35B | 19x |
| **EfficientNet-B4** | **83.0%** | **96.3%** | **19M** | **1x** | **4.2B** | **1x** |
| SENet (Hu et al., 2018) | 82.7% | 96.2% | 146M | 7.7x | 42B | 10x |
| NASNet-A (Zoph et al., 2018) | 82.7% | 96.2% | 89M | 4.7x | 24B | 5.7x |
| AmoebaNet-A (Real et al., 2019) | 82.8% | 96.1% | 87M | 4.6x | 23B | 5.5x |
| PNASNet (Liu et al., 2018) | 82.9% | 96.2% | 86M | 4.5x | 23B | 6.0x |
| **EfficientNet-B5** | **83.7%** | **96.7%** | **30M** | **1x** | **9.9B** | **1x** |
| AmoebaNet-C (Cubuk et al., 2019) | 83.5% | 96.5% | 155M | 5.2x | 41B | 4.1x |
| **EfficientNet-B6** | **84.2%** | **96.8%** | **43M** | **1x** | **19B** | **1x** |
| **EfficientNet-B7** | **84.4%** | **97.1%** | **66M** | **1x** | **37B** | **1x** |
| GPipe (Huang et al., 2018) | 84.3% | 97.0% | 557M | 8.4x | - | - |

**EfficientNet Performance Results Picture source: (Borad)**

This compound scaling method consistently improves model accuracy and efficiency for scaling up existing models such as mobileNet (+1.4% imagenet accuracy), and ResNet (+0.7%), compared to conventional scaling methods.

On comparing EfficientNets(B0-B7) with other models ( ResNets, DenseNets, Xception, Inception, GPipe, e.t.c) there is a significant positive difference in performance ratio, tipping the advantage scale towards EfficientNets.

### 4.4 Evaluation

Using our pretrained model (tez), we can make predictions. A way to accomplish this is to take the test data and run it through the model. This is done by setting the data of the training set to the first test image and then using tez model as my pre-trained model for my data, then I used efficientnet as my classification model.

The figure 4.4 below shows the evaluation processes:

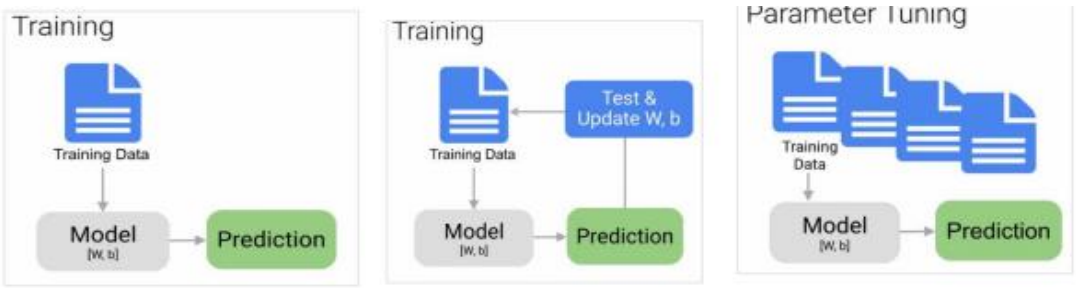


**Figure 4.4 Training the Neural Network**

#### 4.4.1 Evaluation

Using our pretrained model (tez), we can make predictions. A way to accomplish this is to take the test data and run it through the model(tez). This is done by setting the data of the training set to the first test image and then using "tez" model as my pre-trained model for my data, then I used Efficient-Net as my classification model. Image augmentations on this work were done using various transformations on the training dataset, including vertical and horizontal flip, random crop, rotation, height, width, contrast, and zoom. For both the training set and valid set. Flipping and cropping can act as a regulariser to have better performance on the training dataset.

Testing the performance of my model, I used classification accuracy as well as logarithmic loss to evaluate how well our model performed.

# CHAPTER FIVE

# EXPERIMENTS AND RESULTS

The code below illustrates the use of the models:

```
# Importing all important libraries that we need.

train_aug = albumentations.Compose([
            albumentations.RandomResizedCrop(256, 256),
            albumentations.Transpose(p=0.5),
            albumentations.HorizontalFlip(p=0.5),
            albumentations.VerticalFlip(p=0.5),
            albumentations.ShiftScaleRotate(p=0.5)
])

valid_aug = albumentations.Compose([
            albumentations.CenterCrop(256, 256, p=1.0),
            albumentations.Resize(256, 256),
            albumentations.Normalize(
                mean=[0.485, 0.456, 0.406],
                std=[0.229, 0.224, 0.225],
                max_pixel_value=255.0,
                p=1.0)],
            p=1.
)

train_dataset = ImageDataset(
    image_paths= train_image_paths,
    targets=train_targets,
    augmentations= train_aug
)
valid_dataset = ImageDataset(
    image_paths= valid_image_paths,
    targets=valid_targets,
    augmentations= valid_aug
)

# Then reading the csv file
```

```
dataset_read = pd.read_csv ("../input/cassava-leaf-disease-classification/train.csv")

# Splitting dataset in a stratified manner;

df_train, df_valid = model_selection.train_test_split(
dataset_read,
    test_size = 0.1,
    random_state = 50,
    stratify = dataset_read.label.values
)


# to restore index if it has been displaced

df_train = df_train.reset_index(drop= True)
df_valid = df_valid.reset_index(drop= True)

# Getting the training path for our dataset

image_path = "../input/cassava-leaf-disease-classification/train_images"

train_image_paths = [
    os.path.join(image_path, a ) for a in df_train.image_id.values ]

valid_image_paths = [
    os.path.join(image_path, a ) for a in df_valid.image_id.values ]



#Add augmentations:


train_aug = albumentations.Compose([
            albumentations.RandomResizedCrop(256, 256),
            albumentations.Transpose(p=0.5),
            albumentations.HorizontalFlip(p=0.5),
            albumentations.VerticalFlip(p=0.5),
            albumentations.ShiftScaleRotate(p=0.5)
            ]
)
valid_aug = albumentations.Compose([
            albumentations.CenterCrop(256, 256, p=1.0),
            albumentations.Resize(256, 256),
            albumentations.Normalize(
                mean=[0.485, 0.456, 0.406],
                std=[0.229, 0.224, 0.225],
                max_pixel_value=255.0,
                p=1.0
```

```
            )], p=1.)

train_dataset = ImageDataset(
image_paths= train_image_paths,
    targets=train_targets,
    augmentations= train_aug
)
valid_dataset = ImageDataset(
image_paths= valid_image_paths,
    targets=valid_targets,
    augmentations= valid_aug
)

# Model used:

class LeafModel(tez.Model):
    def __init__(self, num_classes):
       super().__init__()
       self.effnet = EfficientNet.from_pretrained("efficientnet-b4")
       self.dropout = nn.Dropout(0.1)
       self.out = nn.Linear(1792, num_classes)
       self.step_scheduler_after = "epoch"
    def monitor_metrics(self, outputs, targets):
        if targets is None:
            return
        outputs = torch.argmax(outputs, dim=1).cpu().detach().numpy()
        targets = targets.cpu().detach().numpy()
        accuracy = metrics.accuracy_score(targets, outputs)
        return {"accuracy": accuracy}
    def fetch_optimizer(self):
        opt = torch.optim.Adam(self.parameters(), lr=3e-4)
        return opt
    def fetch_scheduler(self):
       sch = torch.optim.lr_scheduler.CosineAnnealingWarmRestarts(
            self.optimizer,
            T_0=10, T_mult=1, eta_min=1e-6, last_epoch=-1
       )
       return sch

def forward(self, image, targets=None):
   batch_size, _, _, _ = image.shape
        x = self.effnet.extract_features(image)
        x = F.adaptive_avg_pool2d(x, 1).reshape(batch_size, -1)
        outputs = self.out(self.dropout(x))

        if targets is not None:
            loss = nn.CrossEntropyLoss()(outputs, targets)
            metrics = self.monitor_metrics(outputs, targets)
```

```
            return outputs, loss, metrics
        return outputs, None, None

#Training Model:


es = EarlyStopping(
    monitor="valid_loss",
    model_path="model.bin",
    patience=5, mode="min"
)

model.fit(
    train_dataset,
    valid_dataset=valid_dataset,
    train_bs=32,
    valid_bs=64,
    device="cuda",
    epochs=10,
    callbacks=[es],
    fp16=True,
)
model.save("model.bin")
```

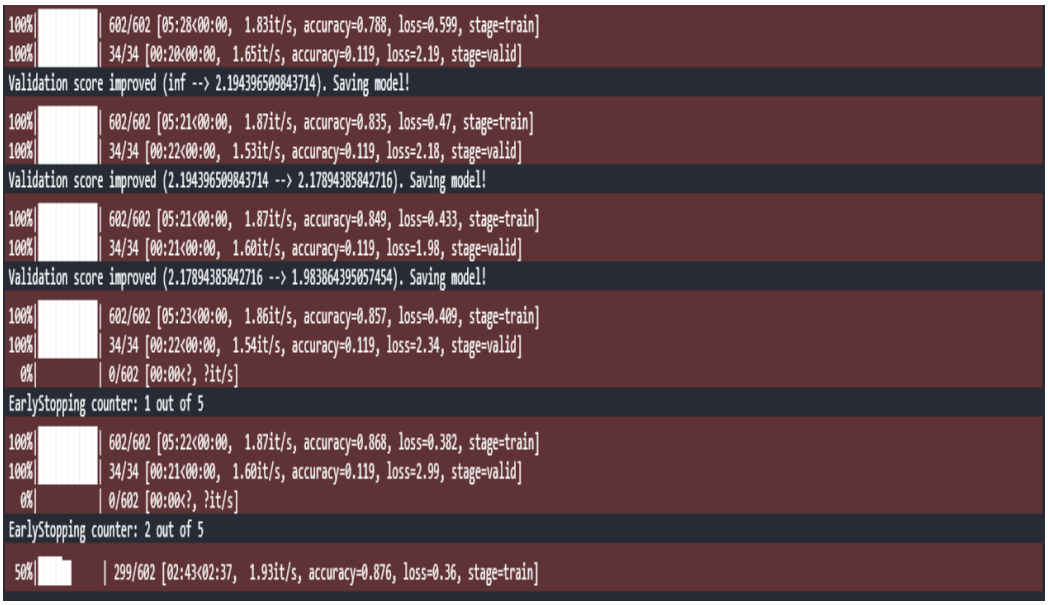


**Figure 5.1 (Performance Metrics: Classification Accuracy and logarithmic loss)**

On deployment of this project, figure 5.1 displays the results gotten from the model, initially only CNN was used, but the accuracy at that point was below 0.6, then in the bid to increase the accuracy

of our model, I searched and read up on the use of EfficientNets on CNN's, on implementing it on the codes, there was a slow but progressive increase in performance accuracy and validation score.

Validation results for the model are provided in two ways and results are presented for the four disease classes studied in the field. First, precision and recall results with an 80–20 training-testing data split, The accuracy is calculated as the percent of the examples the model correctly detects i.e., the proportion of the observations where the predicted and ground truth annotations match. First training of 10 epochs and a patience level of 5, at patience level 1 CA= 0.7889, loss= 0.599 at the training stage, and valid stage, CA= 0.199 & loss = 2.19, at patience level 2, CA= 0.835, loss= 0.47 at training stage, and at valid stage, CA=0.119 & loss=2.18, at patience level 3, CA=0.849, loss =0.43 at training stage, and at valid stage, CA=0.119 & loss=1.98. These results show that the model maintains its average precision for pronounced symptoms in real world images and video and there is a small drop in performance for mild symptoms. It also shows the increase in accuracy and reduction in loss for both the valid and training stages respectively.

With respect to precision, the EfficientNet, CNN model does slightly better on Cassava Mosaic Disease (CMD), label (3), symptoms and slightly worse on Cassava Bacterial Blight (CBB), label (0). And kind of neutral on Cassava Brown Streak Disease (CBSD), label (1), and Cassava Green Mottle (CGM), label (2), Healthy cassava, label (4). We have found that the network learns to generalize as the number iterations increases and so also the accuracy of predictions.

## GITLAB REPOSITORY:[10]

[10] https://gitlab.com/beffiong/cassava-leaf-disease-classification-with-deep-learning/-/tree/master Accessed 15-07-2021

# CHAPTER SIX

# DISCUSSION AND RECOMMENDATIONS

The accuracy of the model could be improved with domain adaptation algorithms to reduce this performance gap in CA. The leaf distortion that occurs in severe CMD infection makes identification of that class straightforward, but where symptoms are mild and leaf distortion was absent, there was a high level of confusion with CBSD (false positive rate is high for CBSD). This is unsurprising, as I have come to realize from different published works, the difficulty of distinguishing between mild symptoms of these two diseases is a common problem faced in real-world field situations by cassava researchers. In this study, we evaluate the performance of a CNN model deployed offline on a real time on a laptop, to detect foliar symptoms of cassava diseases. Accuracy results reflected the decrease in performance moving from real world images. In order to obtain higher accuracy detections, there are a number of potential solutions that can be employed on the CNN model like domain adaptation algorithms to improve performance.

Conclusively, Plant disease tends to be concept of human experience rather than a purely mathematical definition, it is a kind of natural disaster that affects the growth of plants and causes plant death in most cases. the possibility of misidentifying and misclassifying plant disease with a high error rate, as well as the labor-intensive nature to identify and categorize diseases by the artificial eyes, gave rise to the proposed method of using deep learning-based methods to identify plant disease. Disease is an important factor in determining the yield of plants, and as it is well known, the area of agriculture is a very important field to man, with it being the center and core of food production worldwide. It is due to the impact of the agricultural field to man, that Artificial Intelligence (AI) tries to find more ways to make life easier for man and classify plant disease. Convolutional neural network (CNN) models have the potential to improve plant disease phenotyping where the standard approach is visual diagnostics requiring specialized training, This work seeks to help farmers in easily and speedily identifying plant diseases on time, so infected plants can be treated immediately by using our model.

# Bibliography


Adebayo. Segun, Ozichi Emuoyibofarhe, Adebanji Ayandiji, Justice O. Emuoyibofarhe, Oreoluwa James, Oloyede Demeji,Segun Adebayo,. "Detection and Classification of cassava disease using machine learning." *International Journal Of Computer Science And Software Engineering* 8 (2019): 166-176. <https://www.researchgate.net/publication/336020567>.

Albert C.Cruz, Luvisi Andrea, De Bellis Luigi., Ampatzidis Yiannis. "Vision-based plant disease detection system using transfer and deep learning." *ASABE Annual International Meeting*. WA, USA.: Spokane, 2017. p. 1.

Alsayed, Amara Jihen & Bouaziz Bassem & Algergawy. "A Deep Learning-based Approach for Banana Leaf Diseases Classification;." *Proceedings of the BTW (Workshops)*. Stuttgart, Germany: BTW, 2017. pp. 79–88.

Alypaydin, E. *Introduction To machine learning*. Cambridge Massachusetts : MIT Press , 2014.

Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions." *Big Data* (2021): 8, 53. < https://doi.org/10.1186/s40537-021-00444-8>.

Bengio, Yoshua. *Learning Deep Architectures for AI. Foundations.* Hanover: Now Publishers, 2009.

Bishop, Christopher. *Pattern Recognition and machine learning*. New York: Springer, 2006.

Borad, Anand. "Image Classification with EfficientNet: Better performance with computational Efficiency." *Data Monje* (2019): 1-4. <https://datamonje.medium.com/image-classification-with-efficientnet-better-performance-with-computational-efficiency-f480fdb00ac6>.

Brahimi M., Mahmoudi S., Boukhalfa K., Moussaoui A. "Deep interpretable architecture for plant diseases classification." *arXiv.* (2019): 1905. <http://dblp.org/rec/journal/corr/abs-1905-13523.bib>.

Choras, Ryszard S. "Image feature extraction techniques and their applications for CBIR and biometrics systems." *International Journal of Biology and Biomedical Engineering 1* (2007).

DeChant C., Wiesner-Hanks T., Chen S., Stewart E.L., Yosinski J., Gore M.A., Nelson R.J., Lipson H. "Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning." *Phytopathology.* (2017): 1426–1432.

Deng, L. and Yu, D. "Deep Learning: Methods and Applications." *Foundations and Trends in Signal Processing,* (2013): 197-387. <https://doi.org/10.1561/2000000039>.

Dhiraj k. "10 reasons why PyTorch is the deep learning framework of the future." *heartbeat.fritz.ai* (2019)): 1-2.

Erickson, B.J., Korfiatis, P., Akkus, Z. et al. "Toolkits and Libraries for Deep Learning." *J Digit Imaging* (2017): 400–405. <https://doi.org/10.1007/s10278-017-9965-6>.

Fuentes A., Yoon S., Kim S., Park D. “A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition.” *Sensors* (2017): 17.

Ghosal S., Blystone D., Singh A.K., Ganapathysubramanian B., Singh A., Sarkar S. “An explainable deep machine vision framework for plant stress phenotyping.” *Proc. Natl. Acad.Sci* (2018): 4613–4618.

Hadsell, Raia & Sermanet, Pierre & Ben, Jan & Erkan, Ayse & Scoffier, Marco & Kavukcuoglu, Koray & Muller, Urs & Lecun, Yann. “Learning Long-Range Vision for Autonomous Off-Road Driving.” *Journal of field robotics* 26.2 (2009): 26. 120-144. 10.

K. A. Garrett, S. P. Dendy, E. E. Frank, M. N. Rouse, and S. E. Travers. ““Climate change effects on plant disease: genomes to ecosystems.” *Annu Rev Phytopathol* (2006): 489-509.

kapil, Divakar. “Stochastic VS Batch gradiant descent.” *medium Blog* (2019): 1.

LeCun, Y. , Alfredo Canziani. *Deep learning lecture Notes*. new york: new york University, 2020.

LeCun, Y., Bengio, Y. & Hinton, G. “Deep learning.” *Nature* (2015): 436–444. <https://doi.org/10.1038/nature14539>.

Leon Bottou, Olivier Chappelle, Dennis DeCoste, Jason Weston. “"Scaling learning algorithms towards AI" in large scale kernel machines.” *The Institute of electrical and electronics engineers* (2007): 321-359.

Liu, J., Wang, X. “Plant diseases and pests detection based on deep learning.” *Plant Methods 17* (2021): 22.

Lu Y., Yi S., Zeng N., Liu Y., Zhang Y. “Identification of rice diseases using deep convolutional neural networks.” *Neurocomputing* (2017): 378-384.

Ma J., Du K., Zheng F., Zhang L., Gong Z., Sun Z. “A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network.” *computer Electron. Agric.* (2018): 18–24.

Mariyappan., Brindha. “Crop Leaves Disease Identification Using SVM and K-means clustering,.” *Research Gate* (2020): 2.

Murphy, Kelvin P. *Machine learning: a probabilistic perspective*. Cambridge, Massachusetts: MIT press, 2012.

Quoc, Mingxing Tan and. “EfficientNet: Rethinking Model Scaling For Convolutional Neural Networks.” *ICML.2019*. long Beach: CoRR, 2019.

Ramcharan A., Baranowski K., McCloskey P., Ahmed B., Legg J., Hughes D.P. “Deep learning for image-based cassava disease detection.” *Front. Plant Sci.* (2017): 1852.

Saleem, Muhammad Hammad et al. “Plant Disease Detection and Classification by Deep Learning.” *Plants* 8, 11 (2019): 468. <doi:10.3390/plants8110468>.

Shahid, Armughan. “EfficientNet: Scaling of convolutional neural networks done right.” *Towards data science Blog* (2020): 1-2.

Sibiya M., Sumbwanyambe M.A. “Computational Procedure for the Recognition and Classification of Maize Leaf Diseases Out of Healthy Leaves Using Convolutional Neural Networks.” *AgriEngineering* 1 (2019): 119–131. <doi:10.3390/agriengineering1010009>.

Tang, J., Alelyani, S., & Liu, H. “Feature selection for classification: A review. In Data Classification: Algorithms and Applications.” *Classification: Algorithms and Applications* (2014): (pp. 37-64). <https://doi.org/10.1201/b17320>.

Thorpe SJ, Fabre-Thorpe M. “Neuroscience. Seeking categories in the brain.” *Science.* (2001): 291(5502):260-263.

Yann LeCun, Leon Bottou, Yoshua Bengio, Patrick Haffner. “Gradient Based Learning Applied To Document Recognition.” *The Institute of electrical and electronics engineers* (1998): 1-46.