Chapter 1

INTRODUCTION

In India, half the population depends on agriculture for a livelihood. It is the one of the most crucial needs of the life since it supports human and animals with food supply and benefits the human employment and the national economy. Plant diseases result in serious reduction in quality and quantity of agriculture products. Therefore, early detection and diagnosis of these diseases are important. Plant disease detection is a significant challenge in the agriculture sector. Some of the plants show visible symptoms on the plant leaf. These leaf patterns can be used to identify different diseases and take immediate action to prevent the spread. Most of these plant diseases are difficult to detect through naked eyes, and even experienced persons end up wrong. The accuracy of the manual prediction depends upon the experience and knowledge of the person.

The proposed model uses a deep learning model to classify different plant diseases. In standard machine learning models, the features are extracted manually, and the algorithm learns from that data. Thus, it is a two-step process. The deep learning model uses an artificial neural network that can learn features from an input image and make intelligent decisions on its own. Thus, deep learning models are far more capable than the standard machine learning models for image-based classification.

The working model uses convolutional neural networks and transfer learning to classify different plant leaf diseases. CNN is a type of deep learning neural network and has good success in image-based classification. The proposed system is faster and more accurate than the conventional way of manual observation of each plant leaf. This methodology of training facilitates a quick and easy system implementation. In this approach data base is initially created by gathering the crop leaf images and are then used for deep CNN for training and classification. The proposed model effectively classifies between diseased leaves and healthy ones and further It predicts the exact type of the disease.

1.1 Deep Learning

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behaviour of the human brain--albeit far from matching its ability--allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions,

additional hidden layers can help to optimize and refine for accuracy. Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

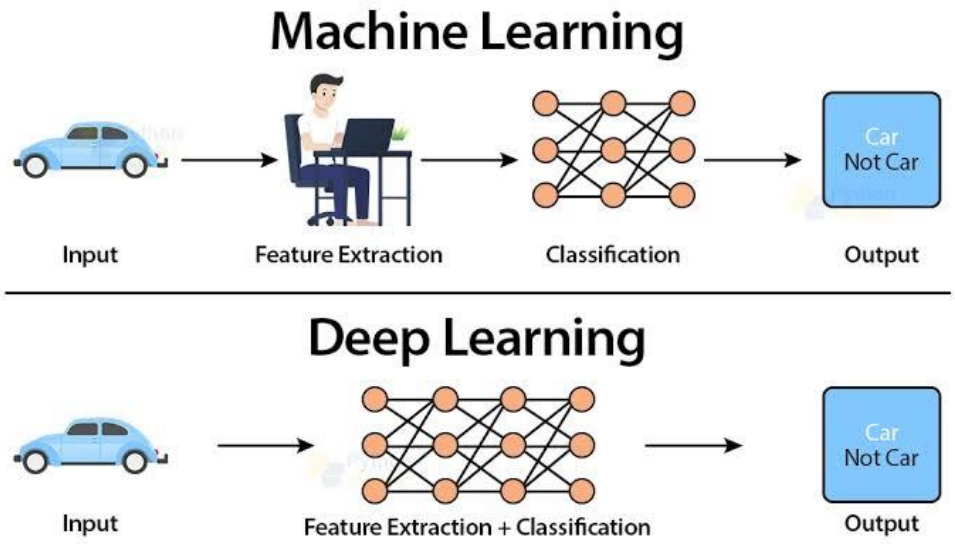

Fig. 1.1 Difference between Machine and Deep Learning

The figure 1.1 shows difference between machine learning and deep learning where feature extraction is done manually in machine learning but in deep learning it is done by the network itself. The network takes the input and gives the output where feature extraction and classification are done internally by the network.

Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, climate science, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance. Most modern deep learning models are based on artificial neural networks, specifically convolutional neural networks (CNN)s, in deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. Deep learning architectures can be constructed with a greedy layer-by-layer method. Deep learning helps to disentangle these abstractions and pick out which features improve performance.

For supervised learning tasks, deep learning methods eliminate feature engineering, by translating the data into compact intermediate representations and derive layered structures that remove redundancy in representation.

Deep learning algorithms can be applied to unsupervised learning tasks. This is an important benefit because unlabelled data are more abundant than the labelled data. Examples of deep structures that can be trained in an unsupervised manner are deep belief networks.

1.2 Convolutional Neural Network

When it comes to machine learning Artificial Neural Networks perform really well. Artificial Neural Networks are used in various classification tasks like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolutional Neural Network. In this Blog, we are going to build basic building block for CNN.

ConvNets derive their name from the "convolution" operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small filters.

There are many important parts of the Convolution Network. These includes the following properties

- Depth: Depth corresponds to the number of filters we use for the convolution operation.
- Stride: Stride is the number of pixels by which we slide our filter matrix over the input matrix.
- Zero-padding: Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix.
- Non-Linearity: ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).
- Spatial Pooling: Spatial Pooling (also called subsampling or down sampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc. These networks have

grown in the number of layers leading to architectures such as RestNet and AlexNet that have been trained on images such as Cifar-10 and then fined tune to other problems, such as plant classification.

1.3 Existing System

The existing method for plant disease detection is simply naked eye observation by experts through which identification and detection of plant diseases is done. For doing so, a large team of experts as well as continuous monitoring of plant is required, which costs very high when we do with large farms.

Leaf shape description is that the key downside in leaf identification. Up to now, several form options are extracted to explain the leaf form. However, there's no correct application to classify the leaf once capturing its image and identifying its attributes, however. In plant leaf classification leaf is classed supported its completely different morphological options. A number of the classification techniques used are:

- Fuzzy logic
- Principal component Analysis
- k-Nearest Neighbours Classifier

1.4 System Overview

The main purpose of proposed system is to detect the diseases of plant leaves. Convolutional neural network (ResNet18), technique is used in classifying the plant leaves into healthy or diseased and if it is a diseased plant leaf, CNN will give the name of that particular disease.

First the images of various leaves are acquired using high resolution camera so as to get the better results & efficiency. The Image is then given to a deep CNN Model (ResNet18) for analysis and the model returns the type of disease as output.

The basic steps of the system are summarized in figure 1.2:

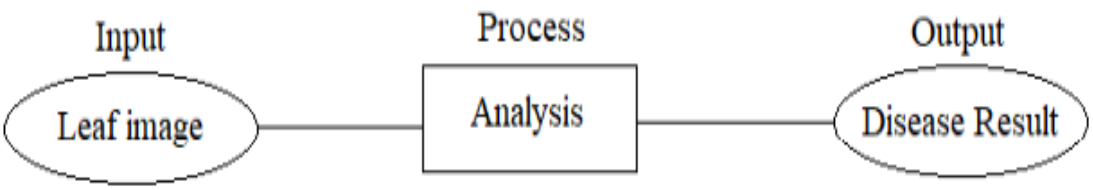

Fig. 1.2 System Overview

1.5 Objectives

The major objectives of this crop disease detection system are as follows:

- To detect the presence of disease on the leaf, and classify into a healthy and diseased leaf.
- To classify the exact type disease based on the symptoms that appear on the leaf.
- To improve the accuracy of multi class classification algorithm in comparison with the traditional methods.

1.6 Purpose

India is an agricultural country, where most of the population depends on agricultural products. So, the cultivation can be improved by technological support. Diseases may cause by pathogen in plant at any environmental condition. In most of the cases diseases are seen on the leaves of the plants, so the detection of disease plays an important role in successful cultivation of crops.

There are lots of techniques to detect the different types of diseases in plants in its early stages. Conventional methods of plant disease detection in naked eye observation methods and it is non-effective for large crops. Using deep learning the disease detection in plant is efficient, less time consuming and accurate. This technique saves time, efforts, labours and use of pesticides. Hope this approach will becomes a little contribution for agriculture sector.

Chapter 2

LITERATURE SURVEY

2.1 Akshai KP, J. Anitha. “Plant disease classification using deep learning”.

The early diagnosis of plant diseases prevents crop loss and the spread of diseases. The CNN model is used to predict different plant diseases correctly. The performance of various pre-trained CNN models such as VGG, ResNet, and DenseNet is observed, and then based on performance metrics, the DenseNet model is found to be more accurate. The model's testing is done using performance evaluation metrics such as accuracy, precision, recall, and F1 score. The DenseNet model achieved the highest accuracy of 98.27%. One of the main problems faced in a larger neural network is the vanishing gradient problem. The ResNet model adds the output of one layer to the next layer, whereas in the DenseNet model, the output feature maps are concatenated with the following future feature maps. Each layer will receive features from the previous layers and pass its features maps to all subsequent layers. Due to that DenseNet model will have features of all complexity and performs well in small datasets. Future works include expand the dataset and increase the number of classes. Another future work is deploying the model into a website/application as it will help farmers/pathologists to identify different diseases using their mobile cameras.

2.2 Ebrahim Hirani, Varun Magotra, Jainam Jain and Pramod Bide “Plant Disease Detection Using Deep Learning”

Convolutional Networks have played a major role in the past for all tasks related to image processing whether the task is related to classification, augmentation, description. But The results of our study show us how visual transformers are the next best thing in computer vision Though transformers were introduced for accommodating the task of Natural language processing, recent studies have shown their potential application in computer vision tasks. For our application which requires the model to be available on minimal resources, the small transformer network (STN) seems to be the best fit. The large transformer network is also a great fit though it dwarfs the STN model when taking size into consideration, it gives the highest accuracy in our study. The proper application of transformers in computer vision tasks needs more research as this architecture is in its early days.

Three approaches are used for the purpose for Plant disease Identification: CNN: A

customized Convolutional neural network is created with 3 convolutional layers followed by relu activation and 3 MaxPooling2D layers (a max pooling layer succeeding each convolutional layer). The input image size that is fed to the network is same as that of the original image (256*256). Different filter sizes are used preceded by a normalization layer and every repeated block of attention followed by feedforward has a residual connection with the previous one. 2 different sizes for transformer models were used: Small Transformer Network (STN): Model with token embedding representations having 256 dimensions, and feedforward layers also outputting 256 dimensions following each attention block. There are 8 attention blocks and input for each block is fed with 8 heads. This model has 3,499,046 parameters which is very less compared to the other models. Large Transformer Network (LTN): This model has token embedding representation of 128 dimensions, with feedforward layers following the attention blocks outputting 128 dimensions. It has 4 attention blocks and each block is fed 4 heads. The model has only 549,926 parameters which is the least of any model that we have tested in this study.

2.3 Achyut Morbekar, Ashi Parihar, Rashmi Jadhav. "Crop Disease Detection Using YOLO".

The YOLO (You Only Look Once) algorithm to detect diseased crops. Crop disease detection system consists of dataset collection, augmentation, object detection, and postprocessing stages.

Managing the crops in the field and saving them from diseases is a major challenge for the farming community. Recent developments and advancements in technology have made it easier to detect crop disease using deep learning. Currently, recent technical models fail to provide high and accurate results in real-time environments. Motivated by this and previous work, the system is proposed to overcome the limitations and provide a fast and efficient automated crop disease detection system for farmers. In this paper, a Plant Village dataset is used to cover possible diseases. This dataset contains numerous images of healthy and diseased crops in a variety of angles and positions. Insufficient images of a few crops lead to overfitting. To evade this issue, several augmentation methods are used. The YOLO deep learning algorithm is used for disease detection, which attains maximum accuracy than other models. The proposed system can be deployed as an aid for modern days farmer for automatic crop disease detection anytime, anywhere. Future work should be focused on detecting diseases on various parts of the crop and evolving phases of the disease.

2.4 M. Akila and P.Deepan, "Detection and Classification of Plant Leaf Diseases by using Deep Learning Algorithm,"

Crop protection in organic agriculture is not a simple matter. It depends on a thorough knowledge of the crops grown and their likely pests, pathogens and weeds. In the system specialized deep learning models were developed, based on specific convolutional neural networks architectures, for the detection of plant diseases through leaves images of healthy or diseased plants. The detector applied images captured in-place by various camera devices and also collected from various resources. The experimental results and comparisons between various deep-architectures with feature extractors demonstrated how our deep-learning-based detector is able to successfully recognize different categories of diseases in various plants and also give solution for concern diseases. Pests/diseases are generally not a significant problem in organic systems, since healthy plants living in good soil with balanced nutrition are better able to resist pest/disease attack. We hope our proposed system will make a suggestive contribution to the agriculture research.

2.5 K. Elangoran, S. Nalini, 2011 “Detection and classification of leaf diseases using K- means-based segmentation and neural-networks-based classification.”

The model uses a concept of plant disease classification using image segmentation and SVM techniques. Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. This paper describes an image processing technique that identifies the visual symptoms of plant diseases using an analysis of coloured images, work of software program that recognizes the colour and shape of the leaf image. LABVIEW software was used to capture the image of plant RGB colour model and MATLAB software is used to enable a recognition process to determine the plant disease through the leaf images. LabVIEW is a graphical programming environment engineers use to develop automated research, validation, and production test systems. The colour model respectively was used to reduce effect of illumination and distinguish between leaf colours efficiently and the resulting colour pixels are clustered to obtain groups of colour in the images.

2.6 Sandesh Raut, Karthik Ingale, "Review on leaf disease detection using Image Processing techniques."

The proposed system is a fast and accurate method for detection and classification of plant diseases. The proposed algorithm is tested on main five diseases on the plant they are Early Scorch, Cottony mold, Ashen Mold, Late scorch, Tiny Whiteness. Initially the RGB image is acquired then a colour transformation structure for the acquired RGB leaf image is created. After that colour value in RGB converted to the space specified in the colour transformation structure. In the next step, the segmentation is done by using K-means clustering technique after that mostly green pixels are masked. Finally, the feature extracted was recognized through a pre-trained neural network. The result show that the proposed system can successfully detect and classify the diseases with a precision between 83% and 94%.

2.7 Sagar Patil, Anjali Chandavale, "A Survey on Methods of Plant Disease Detection"

This survey mainly concentrates on disease detection of dicot plants, here the image acquisition is done by taking RGB image pattern as input and transform it into HSI form, after that for texture analysis CCM and SGDM is used. Stochastic Gradient Descent with Momentum or SGDM is method which helps accelerate gradients vectors in the right directions, thus leading to faster converging. It is one of the most popular optimization algorithms and many state-of-the-art models are trained using it. In agricultural field, rice cultivation plays a vital role. But their growths are affected by various diseases. There will be decrease in the production if the diseases are not identified at an early stage. The main goal of this work is to develop an image processing system that can identify and classify the various rice plant diseases affecting the cultivation of rice namely brown spot disease, leaf blast diseases and bacterial blight disease. This work can be divided into 2 parts namely-rice plant disease detection and recognition of rice plant diseases. In disease detection, the disease affected portion of the rice plant is first identified using KNN and clustering classifier. After that, in disease recognition the rice plant disease type is recognized using classifiers namely KNN and SVM. SVM is less computationally demanding than KNN and is easier to interpret but can identify only a limited set of patterns. On the other hand, KNN can find very complex patterns but its output is more challenging to interpret.

2.8 Mr. Sanjay Mirchandani1, Mihir Pendse2, Prathamesh Rane3, Ashwini Vedula4 "Plant disease detection and classification using image processing and artificial neural networks."

Researchers proposed detection and classification of plant disease using image processing and artificial neural networks in this paper a software solution for fast accurate and automatic detection and classification of plant disease through image processing identification of disease is key to preventing losses in the quality and quantity of the agricultural product this paper discussed the detection and classification of plant diseases is divided into three steps such as identification of infected object extraction of feature set of the infected leaf images and detection and classification the type of disease using ANN. Different techniques are adopted for detection and diagnosis the disease but the better way is by image processing the author suggested a method in which initially the infected region is found then different features are extracted such as colour texture and shape finally parameter classification technique is used for detecting the diseases.

2.9 Savita N. Ghaiwat, Parul Arora "Detection and Classification of Plant Leaf Diseases Using Image Processing Techniques:"

The model uses an image processing technique for detection and classification of plant disease classification technique deals with classifying each pattern in one of the distinct classes .The author suggested so many techniques for classification such as K nearest neighbour classifier probabilistic neural network genetic algorithm support vector machine and principal component analysis artificial neural network Fuzzy Logic the technique is presented using image processing as a tool to enhance the feature extraction of an image by using of local binary pattern as a parameter.

The local binary pattern approach has evolved to represent a significant breakthrough in texture analysis outperforming earlier method in many application study of image analysis takes which have not been generally considered texture analysis problems is done results suggest that texture analysis and the ideas behind the lbp method could have a much wider role in image analysis and computer vision then was thought before.

2.10 Jayamala k Patil, Rajkumar "Advances in image processing for plant disease detection"

To retrieve the related images the search is done in Two Steps, first step is matches the images by comparing the standard deviations for three colour components the second step is weighted version of the Euclidean distance between the feature coefficients of an image selected in the first step, this reported following important image processing method first one image clipping separating the leaf with spots from the complex background. Noise resolution to filters simple filter and median filter work compared and at last median filter was chosen. Thresholding to segment or partition image into the spot background fourth one segmentation they used OSTU's method. K-means clustering and back propagation feed-forward neural network for clustering and classification. There are two main characteristics of plant disease detection using machine learning method that must be achieved their speed and accuracy hence innovative efficient and fast interpreting algorithms which will help plants scientists in detecting disease work proposed by the researcher can be extended for development of hybrid algorithms such as genetic algorithms and neural networks in order to increase the recognition rate of the final classification process.

Chapter 3

DESIGN METHODOLOGY

3.1 Crop Disease Detection Model Flowchart

A data flow Diagram is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A data flow diagram is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. The below flow chart gives the methodology of crop disease detection model as shown in figure in 3.1

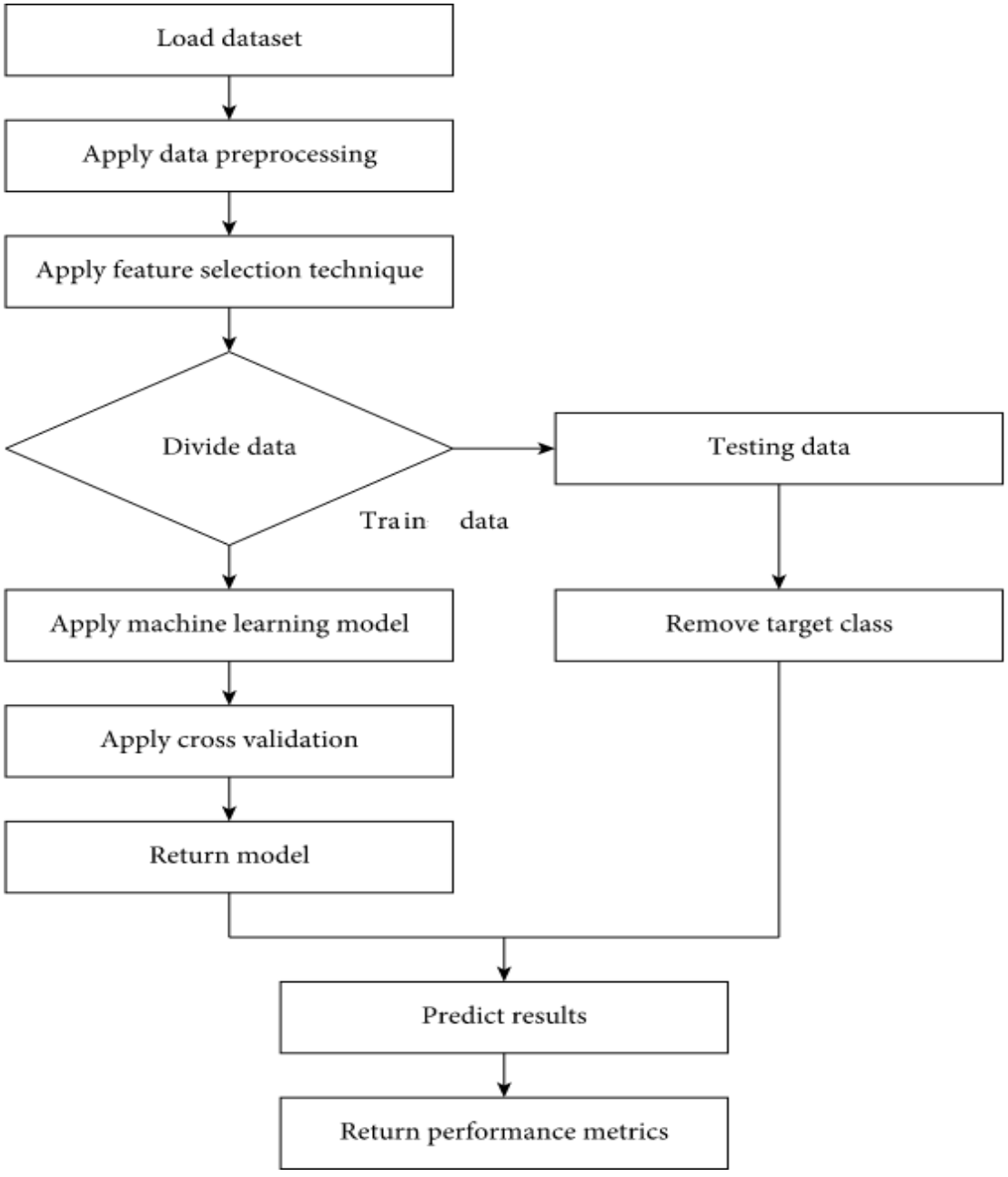

Fig. 3.1 Flowchart of Crop Disease Detection Model

3.1.1 Dataset

The images of various leaves are acquired using high resolution camera so as to get the better results & efficiency. The images of various mango leaves where acquired from our collage and from remote mango farms, out of which four classes were considered namely healthy leaves and three most common diseases observed which are leaf galls, anthracnose and tuberculosis. The below table 3.1 shows the dataset composition and images collected for each class.

Table 3.1 Dataset Composition

Class Name	Image Count
Healthy Leaf	200
Leaf Galls	200
Anthracnose	200
Tuberculosis	200
	TOTAL-800

A total of 800 images were acquired with each of the four classes consisting of 200 images each which was given to the model for training and testing.

3.1.2 Data Preprocessing

Data preprocessing is a process of preparing the raw data and making it suitable for a deep learning model. It is the first and crucial step while creating a deep learning model. When creating a deep learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data preprocessing task.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for deep learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a deep learning model which also increases the accuracy and efficiency of a machine learning model.

It involves below steps:

- Getting the dataset
- Importing libraries

- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

3.1.3 Cross Validation

Cross-validation is a technique for evaluating the model and testing its performance. Cross-validation is commonly used in applied deep learning tasks. It helps to compare and select an appropriate model for the specific predictive modelling problem. Cross-validation is easy to understand, easy to implement, and it tends to have a lower bias than other methods used to count the model's efficiency scores. All this makes cross-validation a powerful tool for selecting the best model for the specific task.

There are a lot of different techniques that may be used to cross-validate a model. Still, all of them have a similar algorithm:

- Divide the dataset into two parts: one for training, other for testing.
- Train the model on the training set.
- Validate the model on the test set.
- Repeat 1-3 steps a couple of times. This number depends on the method.

Dataset is created by capturing live images of various crop diseases. The collected dataset is applied with various data preprocessing techniques. Feature selection technique is applied to extract the required features. Then the dataset is divided into train and test which is 80% and 20% respectively. The training dataset is applied to the model.

Cross validation is performed on the train data set. Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

After the cross validation the model is returned which is used for testing. In testing data, the target classes are removed. After the target classes are removed, the test data is given to the model returned after cross validation and in turn it is used to predict results. Based on the predicted results performance metrics is obtained and used for analysis of various parameters like the model accuracy and loss. The obtained performance metrics is used to analysis if the model is best suited for the task.

3.2 ResNet-18 Architecture

The Crop Disease Detection System uses ResNet-18 Architecture.ResNet18 is a 72-layer architecture with 18 deep layers. The architecture of this network aimed at enabling large amounts of convolutional layers to function efficiently. However, the addition of multiple deep layers to a network often results in a degradation of the output. This is known as the problem of vanishing gradient where neural networks, while getting trained through back propagation, rely on the gradient descent, descending the loss function to find the minimizing weights. Due to the presence of multiple layers, the repeated multiplication results in the gradient becoming smaller and smaller thereby "vanishing" leading to a saturation in the network performance or even degrading the performance. The below figure 3.2 represents Resnet-18 architecture with 18 layers.

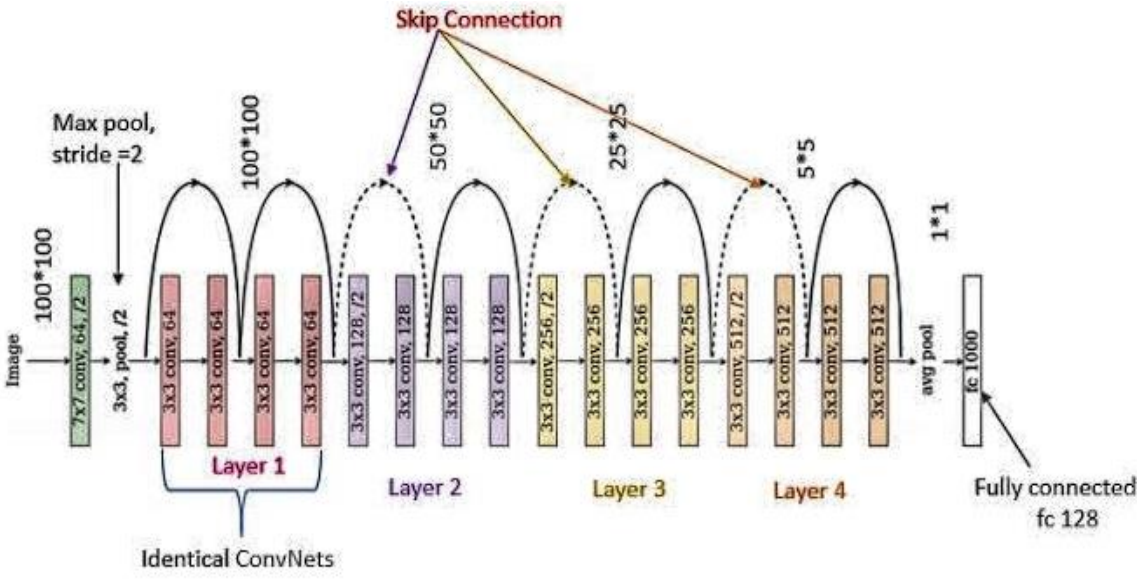

Fig. 3.2 ResNet-18 Architecture

The primary idea of ResNet is the use of jumping connections that are mostly referred to as shortcut connections or identity connections. These connections primarily function by hopping over one or multiple layers forming shortcuts between these layers. The aim of introducing these shortcut connections was to resolve the predominant issue of vanishing gradient faced by deep networks. These shortcut connections remove the vanishing gradient issue by again using the activations of the previous layer. These identity mappings initially do not do anything much except skip the connections, resulting in the use of previous layer activations. This process of skipping the connection compresses the network; hence, the network learns faster. This compression of the connections is followed by expansion of the layers so that the residual part of the network could also train and explore more feature space.

The table 3.2 below represents different convolution layers parameters like output image size, filter size, padding and stride of ResNet-18 Architecture.

Table 3.2 ResNet-18 Architecture Layer Parameters

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

RESNET18 has 18 layers with a 7x7 kernel as 1st layer. It has four layers of ConvNets that are identical. Each layer consists of two residual blocks. Each block consists of two weight layers with a skip connection connected to the output of the second weight layer with a ReLU. If the result is equal to the input of the ConvNet layer, then the identity connection is used. But, if the input is not similar to the output, then a convolutional pooling is done on the skip connection.

The input size taken by the RESNET18 is (224, 224, 3), which is done by applying augmentation using AugStatic library in pre-processing step. In (224, 224, 3) where 224 is the width and height. 3 is the RBG channel. The output is a Fully Connected (FC) layer that gives input to the sequential layer. The fully connected layer is followed by an activation function SoftMax. An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network.

The Architecture consists of the following sequential modules (each one may contain more than one layer)

- Convolution
- ReLu activation function
- Pooling
- Fully connected layers
- Output layer

3.2.1 Convolution

Convolution operation is an element-wise matrix multiplication operation. Convolutional layers take the three-dimensional input matrix mentioned before and they pass a filter (also known as convolutional kernel) over the image, applying it to a small window of pixels at a time (i.e. 3x3 pixels) and moving this window until the entire image has been scanned. The convolutional operation calculates the dot product of the pixel values in the current filter window along with the weights defined in the filter. The output of this operation is the final convoluted image.

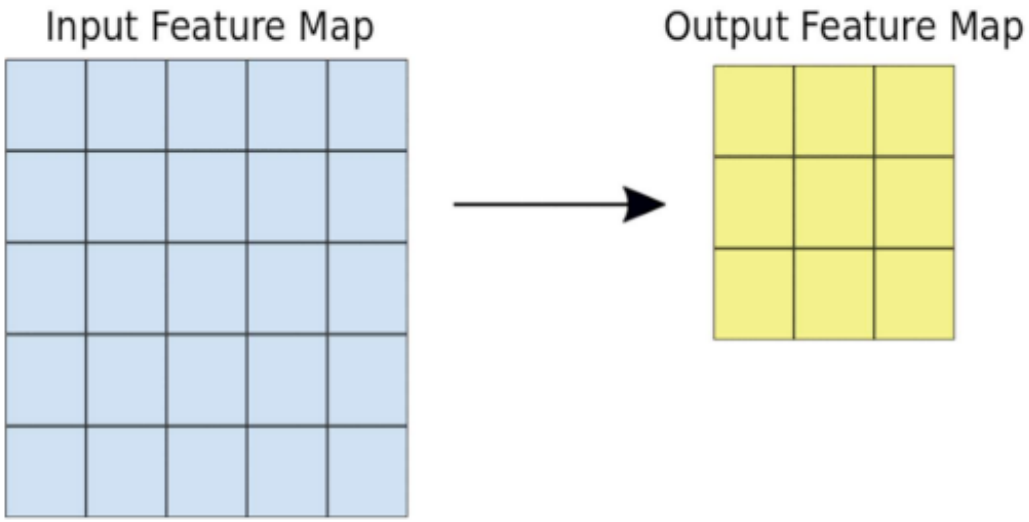

Fig. 3.3 Convolution Overview

The above figure 3.3 shows a basic overview of convolution layer operation. The core of image classification CNNs is that as the model trains what it really does is that it learns the values for the filter matrices that enable it to extract important features (shapes, textures, coloured areas, etc) in the image. Each convolutional layer applies one new filter to the convoluted image of the previous layer that can extract one more feature. So, as it stacks more filters, the more features the CNN can extract from an image.

3.2.2 ReLu activation function

After each convolution operation, CNN applies to the output a Rectified Linear Unit (ReLu) function to the convolved image. ReLu is very commonly used in machine learning and deep learning applications because it introduces nonlinearity into the model. This helps model to generalize better and avoid overfitting. As a consequence, the usage of ReLU helps to prevent the exponential growth in the computation required to operate the neural network. If the CNN scales in size, the computational cost of adding extra ReLU's increases linearly.

3.2.3 Pooling

Pooling is the process where the CNN down samples the convolved image by reducing the number of dimensions of the feature map. It does so to reduce processing time and the computing power needed. During this process, it preserves the most important feature information. There are several methods that can be used for pooling. The most common ones are Max pooling and Average pooling. In this application, max pooling is used as it is the most effective most of the times. The below figure 3.4 shows general max pooling operation.

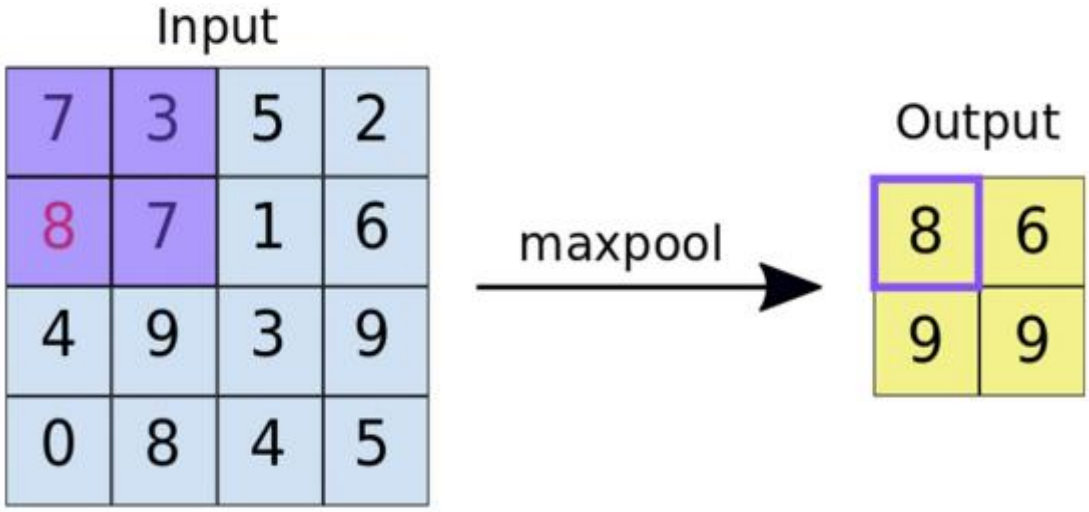

Fig 3.4 Max Pooling

Max pooling is very similar to the convolution process. A window slides over the feature map and extracts tiles of a specified size. For each tile, max pooling picks the maximum value and adds it to a new feature map. Max pooling is done to in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.

3.2.4 Fully Connected Layers

After pooling, there is always one or more fully connected layers. These layers perform the classification based on the features extracted from the image by the previously mentioned convolution processes. Fully Connected Layer is simply, feed forward neural networks. Fully Connected Layers form the last few layers in the network. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

3.2.5 Output Layer

The last fully connected layer is the output layer which applies a SoftMax function to the output of the previous fully connected layer and returns a probability for each class. The SoftMax function, also known as softargmaxor normalized exponential function, converts a vector of K real numbers into a probability distribution of K possible outcomes. It is a generalization of the logistic function to multiple dimensions, and used in multinomial logistic regression. The SoftMax function is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output class. After passing through the fully connected layers, the final layer uses the SoftMax activation function (instead of ReLU) which is used to get probabilities of the input being in a particular class (classification). And so finally, we have the probabilities of the object in the image belonging to the different classes.

Chapter 4

DESIGN IMPLEMENTATION

4.1 Hardware Requirements

- NVIDIA Jetson Nano 4GB Developer Kit
- Logitech C270 Digital HD Camera
- Cooling Fan
- Power Adopter
- Memory Card
- Monitor, USB keyboard and mouse

4.1.1 NVIDIA Jetson Nano 4GB Developer Kit

NVIDIA Jetson Nano Developer Kit is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. All in an easy-to-use platform that runs in as little as 5 watts. The figure 4.1 represents NVIDIA Jetson Nano 4GB Developer Kit.

Fig. 4.1 NVIDIA Jetson Nano 4GB Developer Kit

The developer kit can be powered by micro-USB and comes with extensive input / output systems, ranging from GPIO to CSI. This makes it simple for developers to connect a diverse set of new sensors to enable a variety of AI applications. It’s incredibly power-efficient, consuming as little as 5 watts. Jetson Nano is also supported by NVIDIA Jetpack, which includes a board support package (BSP), Linux OS, NVIDIA CUDA, cuDNN, and

TensorRT™ software libraries for deep learning, computer vision, GPU computing, multimedia processing, and much more. The software is even available using an easy-to-flash SD card image, making it fast and easy to get started.

The same Jetpack SDK is used across the entire NVIDIA Jetson™ family of products and is fully compatible with NVIDIA's world-leading AI platform for training and deploying AI software. This proven software stack reduces complexity and overall effort for developers

The figure 4.2 shows different connections available in NVIDIA Jetson Nano 4GB Developer Kit.

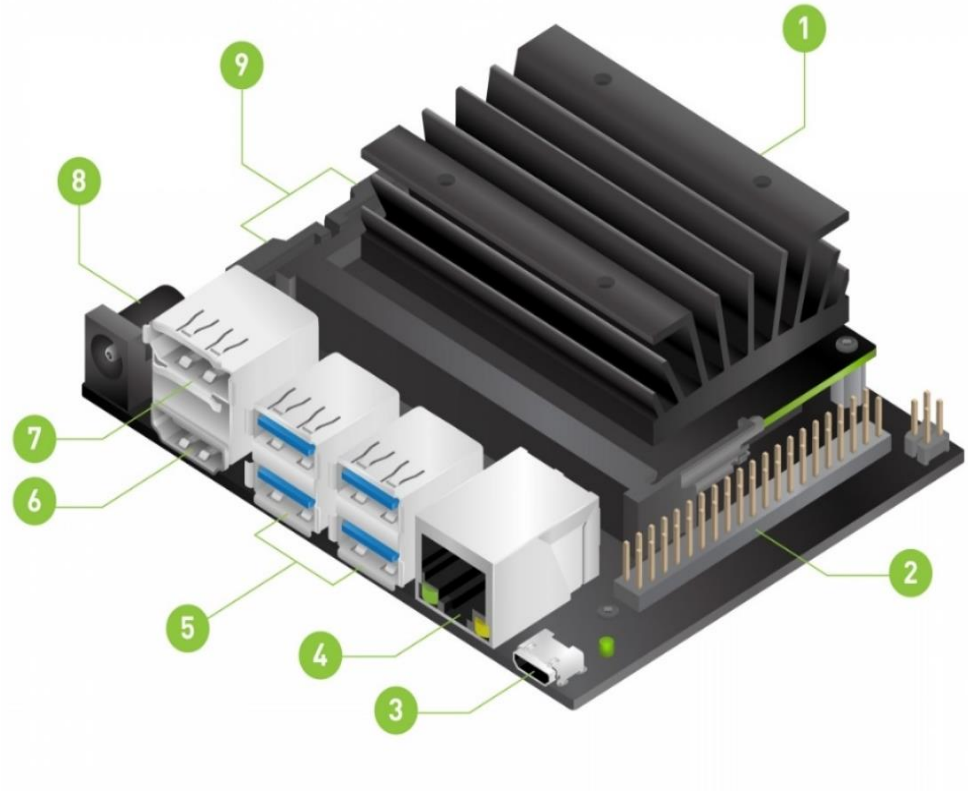

Fig. 4.2 Jetson Nano Developer Kit Connections

1. microSD card slot for main storage
2. 40-pin expansion header
3. Micro-USB port for 5V power input, or for Device Mode
4. Gigabit Ethernet port
5. USB 3.0 ports (x4)
6. HDMI output port
7. DisplayPort connector
8. DC Barrel jack for 5V power input
9. MIPI CSI-2 camera connectors

The Jetson Nano Developer Kit is ideal for deep learning and AI applications, and it comes with a 40-pin GPIO header, that can be used for microcontroller level tasks.

Table 4.1 Technical Specifications of Jetson Nano Developer Kit

DEVELOPER KIT	
GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD (not included)
Video Encoder	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Decoder	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30| (H.264/H.265)
Camera	1x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI 2.0 and eDP 1.4
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I^2C, I^2S, SPI, UART
Mechanical	100 mm x 80 mm x 29 mm

The above table 4.1 specifies the technical specifications of Jetson Nano Developer Kit with key highlights being 128-core Maxwell GPU, Quad-core ARM Cortex-A57 CPU and 4GB LPDDR4 (RAM) Memory.

4.1.2 Logitech C270 Digital HD Camera

Input device to jetson nano is a high-resolution camera used to do real time classification from the camera image input, also the camera is used for capturing images which form the dataset. The figure 4.3 shows Logitech C270 Camera which is used as an input device.

Fig. 4.3 Logitech C270 Digital HD Camera

Logitech C270 HD Camera has a video capture resolution of 720p and one of its key features being it works in low light conditions. The Camera is capable of capturing a Crisp HD 720p/30 fps video with diagonal 55° field of view and auto light correction. The built-in noise-reducing mic makes sure your voice comes across clearly up to 1.5 meters away, even if you're in busy surroundings.C270's Right Light 2 feature adjusts to lighting conditions, producing brighter, contrasted images. The adjustable universal clip lets you attach the camera securely to your screen or laptop, or fold the clip and set the camera on a shelf.

4.1.3 Cooling Fan and Power Adapter

Jetson Nano is a single board computer which can get quite hot while creating or training complex deep leering models. While passive cooling options are often good enough to avoid overheating and thermal throttling, at some point a cooling fan must be used to reduce overheating and increase the performance of Jetson Nano. Jetson Nano requires 5 volts and 4 amperes power supply to provide exact power input the device power adapter is used to avoid damaging the device and provide optimal functioning of jetson nano.

4.1.4 Monitor and Memory Card

Jetson Nano Developer kit supports all display units. The display unit is required to display the output after image classification. A 64Gb memory card is used for storage purpose which is inserted in the memory card slot in the jetson nano.

4.2 Software Requirements

Python is used for coding and in development of the deep learning model because python contains large collection libraries used in deep learning.

4.2.1 Python Programming Language

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional

programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

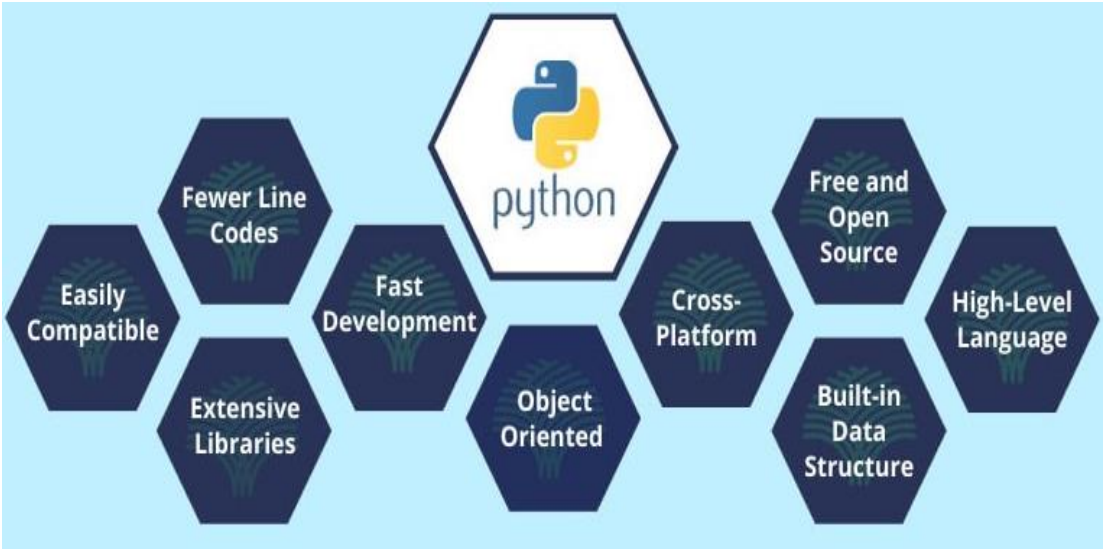

Fig. 4.4 Key Features of Python Programming language

The above figure 4.4 highlights the key features of python programming language which includes Cross-Platform support and large collection of libraries.

4.2.2 Libraries used in development of Deep Learning Model

- **Jetson-Inference:** Jetson-inference is a training guide for inference on the NVIDIA Jetson Nano. To run First, clone the jetson-inference project similar to the Docker Container. Next, to download all the necessary files, and build the project create a folder called build and run cmake. Then, Model-Downloader tool will run automatically on the screen.

  Cloning the Repo

  To download the code, navigate to a folder of your choosing on the Jetson. First, make sure git and cmake are installed:

  $ sudo apt-get update

  $ sudo apt-get install git cmake

  Then clone the jetson-inference project:

  $ git clone https://github.com/dusty-nv/jetson-inference

  $ cd jetson-inference

  $ git submodule update --init

  Remember to run the git submodule update --init step (or clone with the --recursive flag).

Project to create bindings for Python 3.6, install these packages before proceeding:

$ sudo apt-get install libpython3-dev python3-numpy

Installing these additional packages will enable the repo to build the extension bindings for Python 3.6, in addition to Python 2.7 (which is already pre-installed). Then after the build process, the jetson.inference and jetson.utils packages will be available to use within your Python environments.

- Configuring with CMake

Next, create a build directory within the project and run cmake to configure the build. When cmake is run, a script is launched (CMakePreBuild.sh) that will install any required dependencies and download DNN models.

$ cd jetson-inference # omit if working directory is already jetson-inference

$ mkdir build

$ cd build

$ cmake ../

- Downloading Models

The project comes with many pre-trained networks that can you can choose to have downloaded and installed through the Model Downloader tool (download-models.sh). By default, not all of the models are initially selected for download to save disk space. When initially configuring the project, cmake will automatically run the downloader tool. The below figure 4.5 represents model downloader.

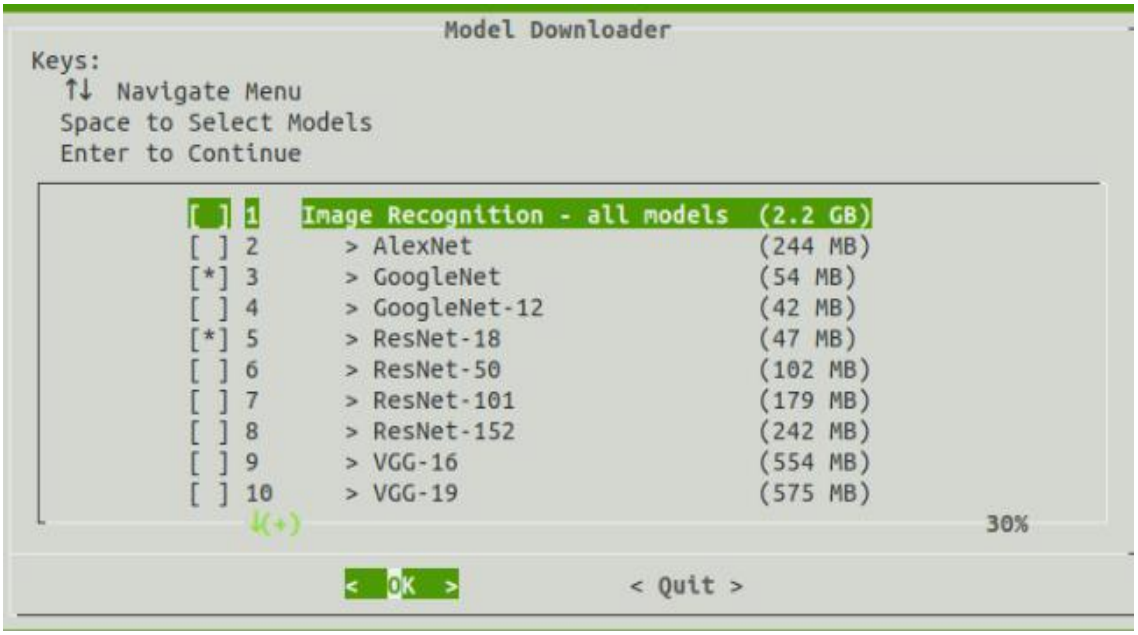

Fig. 4.5 Jetson-Inference Model Downloader

To run the tool again

$ cd jetson-inference/build

$ ./install-pytorch.sh

Compiling the Project: Make sure you are still in the jetson-inference/build directory, created above in step Then run make followed by sudo make install to build the libraries, Python extension bindings, and code samples:

$ cd jetson-inference/build # omit if working directory is already build/ from above

$ make

$ sudo make install

$ sudo ldconfig

- **Jetson-utils:** Jetson-utils is a C++ library typically used in Hardware, GPU, OpenCV applications. jetson-utils has no bugs, it has no vulnerabilities and it has low support. However, jetson-utils has a Non-SPDX License. Can be downloaded from GitHub. C++/Python Linux utility wrappers for NVIDIA Jetson - camera, codecs, CUDA, GStreamer, HID, OpenGL/XGL.

- **OpenCV:** OpenCV is an open-source library which is very useful for computer vision applications such as video analysis, CCTV footage analysis and image analysis. OpenCV is written by C++ and has more than 2,500 optimized algorithms. When we create applications for computer vision that we don't want to build from scratch we can use this library to start focusing on real world problems. We can easily install it in any OS like Windows, Ubuntu and MacOS.
  Run the command below to install the OpenCV library in the Ubuntu system sudo apt **install** python3-opencv

  TO READ IMAGE
  This is Syntax to read the image:
  variable= cv2.imread('path of the image file')

  TO WRITE IMAGE
  Syntax to write image:
  variable= cv2.imwrite('path of the image file')

GET IMAGE INFORMATION

It provides the shape value of an image, like height and width pixel values.

Syntax: img = cv2.imread('photo.jpg')

CAPTURING VIDEO USING OPENCV

OpenCV has a function to read video, which is cv2.VideoCapture(). We can access our webcam using pass 0 in the function parameter. If you want to capture CCTV footage then we can pass RTSP url in the function parameter, which is really useful for video analysis.

Syntax: video = cv2.VideoCapture(0)

- **NumPy:** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

  NumPy (Numerical Python) is an open-source Python library that's used in almost every field of science and engineering. It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystems. NumPy users include everyone from beginning coders to experienced researchers doing state-of-the-art scientific and industrial research and development.

  The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages. The NumPy library contains multidimensional array and matrix data structures (you'll find more information about this in later sections). It provides nd array, a homogeneous n-dimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

  Installing NumPy in Python

  Syntax: pip install numpy

  Importing NumPy in Python

  Syntax: import numpy as np

4.3 Jetson Nano Setup Procedure

The Jetson Nano Developer Kit uses a microSD card as a boot device and for main storage. It's important to have a card that's fast and large enough for projects. The minimum recommended is a 32 GB UHS-1 card. See the instructions below to flash microSD card with operating system and software.

4.3.1 Write Image to the microSD Card

To prepare the microSD card, a computer with Internet connection is needed and the ability to read and write SD cards, either via a built-in SD card slot or adapter. Download the Jetson Nano Developer Kit SD Card Image, and note where it was saved on the computer. Write the image to microSD card by following the instructions below.

Format your microSD card using SD Memory Card Formatter from the SD Association. The below figure 4.6 represents SD Card Formatter Window.

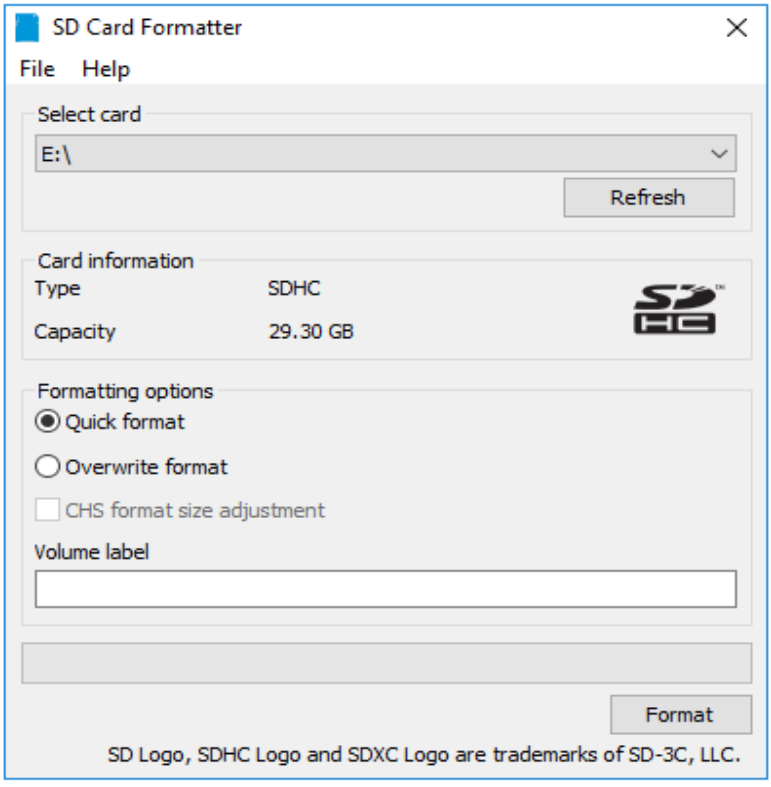

Fig. 4.6 SD Card Formatter Window

- Download, install, and launch SD Memory Card Formatter for Windows.
- Select card drive
- Select "Quick format"
- Leave "Volume label" blank

- Click “Format” to start formatting, and “Yes” on the warning dialog

Use Etcher to write the Jetson Nano Developer Kit SD Card Image to your microSD card

- Download, install, and launch Etcher. Click “Select image” and choose the zipped image file downloaded earlier.
- Insert your microSD card if not already inserted. Click Cancel if Windows prompts with a dialog box.
- Click “Select drive” and choose the correct device. Figure 4.7 shows etcher interface.

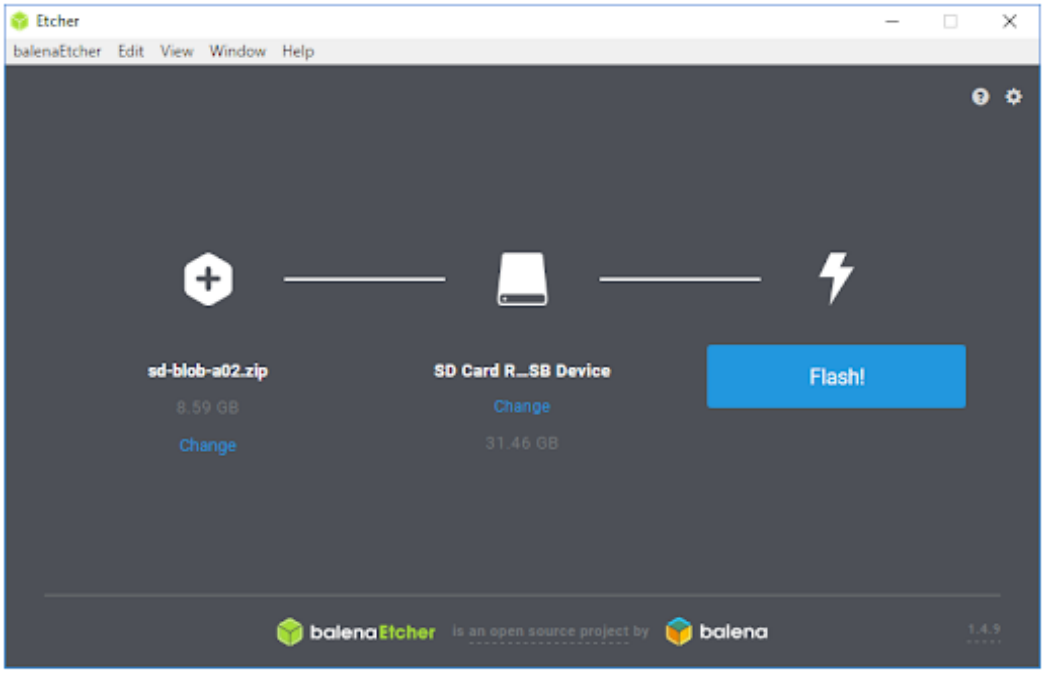

Fig. 4.7 Etcher Interface

- Click “Flash!” It will take Etcher about 10 minutes to write and validate the image if microSD card is connected via USB3.
- After Etcher finishes, Windows may show it doesn’t know how to read the SD Card. Just click Cancel and remove the microSD card.
- After microSD card is ready, proceed to set up your developer kit.

4.3.2 Initial Setup with Display

- Insert the microSD card (with system image already written to it) into the slot on the underside of the Jetson Nano module.
- Set the developer kit on top of the paper stand.
- Power on your computer display and connect it

- Connect the USB keyboard and mouse.
- Connect your Micro-USB power supply. The developer kit will power on and boot automatically.

First Boot: A green LED next to the Micro-USB connector will light as soon as the developer kit powers on. When it is booted for the first time, the developer kit will ask for some initial setup, including:

- Review and accept NVIDIA Jetson software EULA
- Select system language, keyboard layout, and time zone
- Create username, password, and computer name
- Select APP partition size—it is recommended to use the max size suggested

After Logging In: This screen will appear. The figure 4.8 represents jetson nano interface after display setup procedure.

Fig. 4.8 Jetson Nano Display Interface

4.4 Crop Disease Detection System

The system consists of three main units. The input unit to the jetson Nano which camera, Jetson Nano where the model is trained / developed and the output unit which is the monitor which is used for displaying the results after the model has done the predication based on the input image given to jetson nano.

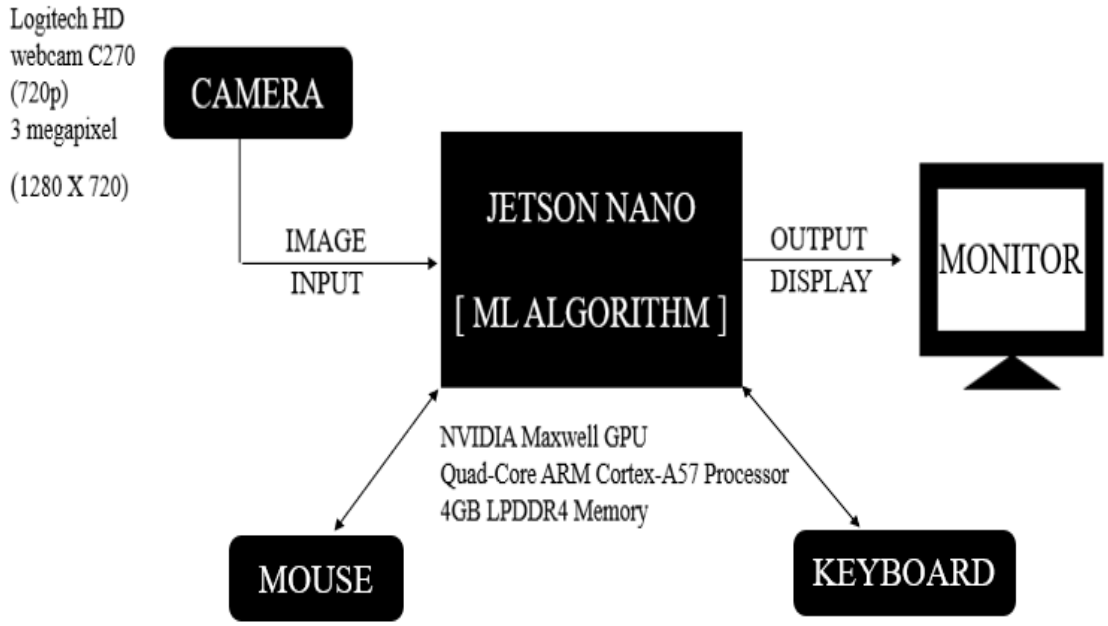

Fig. 4.9 Crop Disease Detection System Block Diagram

The above figure 4.9 represents the block diagram of crop disease detection system which consists of three main units.

- **Input Unit**: The input unit is a Logitech C270 HD Camera which is used to feed the input image to the jetson nano from the live video which used of real time classification. The dimensions of the input image to the jetson nano is 1280 X 720. The Logitech camera captures the video at 720p and 30 frames per second.

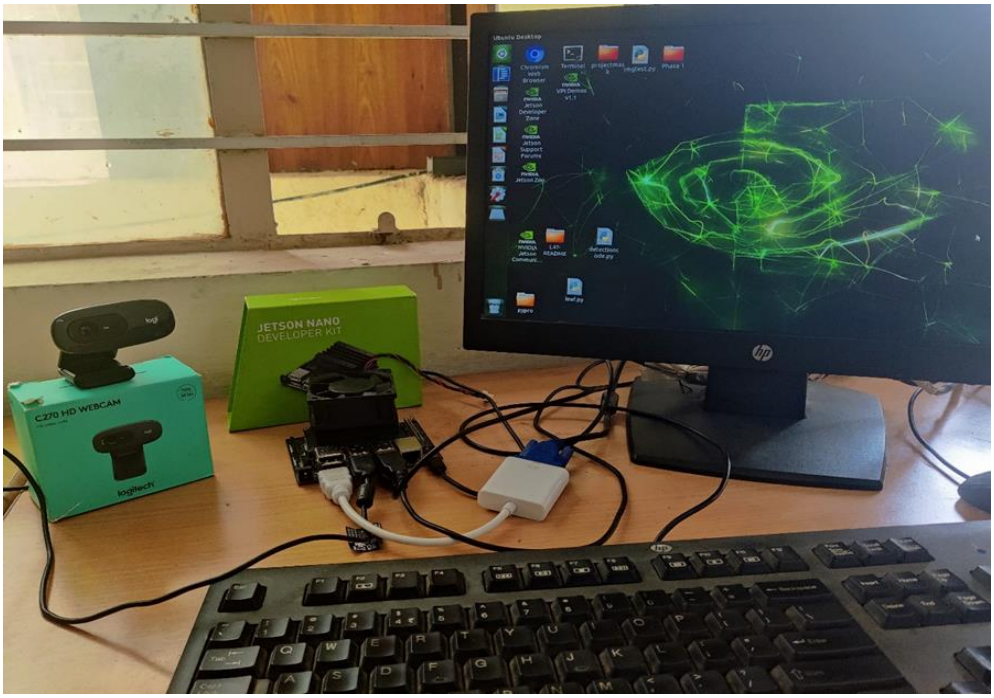

Fig. 4.10 Real Time Crop Disease Detection System

- **Processing Unit**: The processing unit where the machine learning algorithm runs is the Jetson Nano 4GB Developer Kit. Which consists of Maxwell GPU, Quad-Core ARM Cortex-A57 Processor and 4GB LPDDR4 (RAM) memory. The input Jetson Nano Developer Kit is an image from the input unit camera. The deep learning model classifies the image into the four defined classes which are Healthy leaf, Leaf Galls, Anthracnose, Tuberculosis. The predicted results are displayed on a display unit.
- **Output Unit**: The output unit is a monitor used to display the predicted class of the disease given from the jetson nano developer kit after real time classification.
- Other components connected to Jetson nano are cooling fan to avoid overheating of the system, Keyboard and mouse (USB) to operate the display unit and jetson nano.

4.4 Crop Disease Detection System Code

```
import jetson.inference

import jetson.utils

import cv2

import numpy as np

import time

width=1280

height=720

dispW=width

dispH=height

flip=2

#camSet='nvarguscamerasrc !  video/x-raw(memory:NVMM), width=3264, height=2464, format=NV12, framerate=21/1 ! nvvidconv flip-method='+str(flip)+' ! video/x-raw, width='+str(dispW)+', height='+str(dispH)+', format=BGRx ! videoconvert ! video/x-raw, format=BGR ! videobalance  contrast=1.5 brightness=-.3 saturation=1.2 ! appsink  '
```

```
cam1=cv2.VideoCapture("/home/priya/Documents/stem.mp4")

#cam1=cv2.VideoCapture('/dev/video1')

#cam1.set(cv2.CAP_PROP_FRAME_WIDTH,dispW)

#cam1.set(cv2.CAP_PROP_FRAME_HEIGHT,dispH)

net=jetson.inference.imageNet('resnet18',['--model=/home/priya/Downloads/jetson-inference/python/training/classification/myModel/resnet18.onnx','--input_blob=input_0','--output_blob=output_0','--labels=/home/priya/Downloads/jetson-inference/mytrain/labels.txt'])

font=cv2.FONT_HERSHEY_SIMPLEX

timeMark=time.time()

fpsFilter=0

while True:

    _,frame=cam1.read()

    img=cv2.cvtColor(frame,cv2.COLOR_BGR2RGBA).astype(np.float32)

    img=jetson.utils.cudaFromNumpy(img)

    classID, confidence =net.Classify(img, width, height)

    item="

    item =net.GetClassDesc(classID)

    dt=time.time()-timeMark

    fps=1/dt

    fpsFilter=.95*fpsFilter +.05*fps

    timeMark=time.time()
```

```
    cv2.putText(frame,str(round(fpsFilter,1))+' fps '+item,(0,30),font,1,(0,0,255),2)
    cv2.imshow('recCam',frame)
    cv2.moveWindow('recCam',0,0)
    if cv2.waitKey(1)==ord('q'):
        break
cam.release()
cv2.destroyAllWindows()
```

Chapter 5

RESULTS

5.1 General Output Flow

The below figure 5.1 shows the general output flow in crop disease detection system. The Image input to the Jetson Nano is taken from the live video footage of the camera.

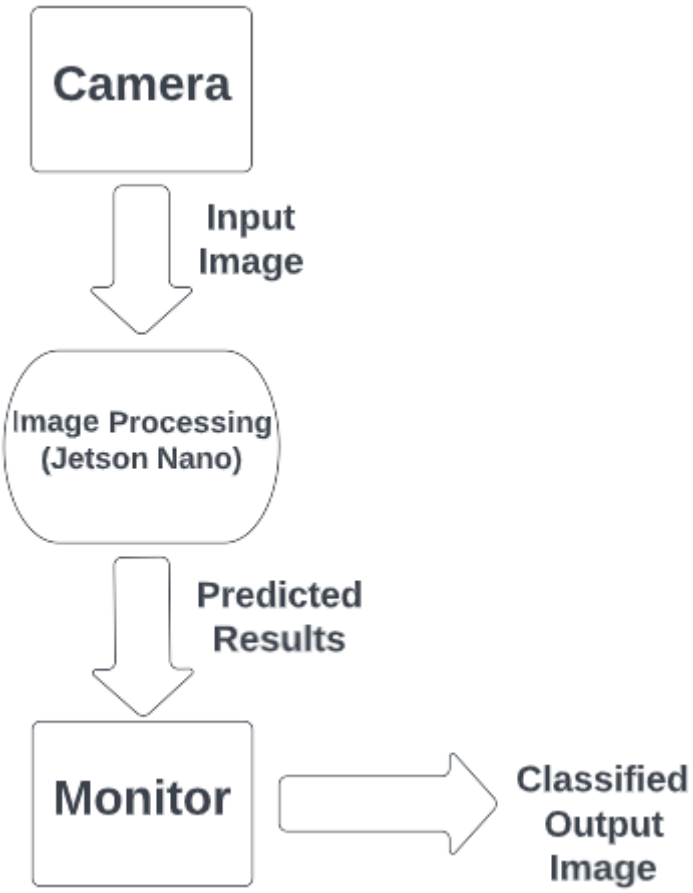

Fig 5.1 General Output Flow Diagram

Once the image is given as an input the jetson nano developer kit, it performs image processing and the deep learning model predicted the results and it is given to a display unit monitor which displays the classified output image, which shows if the leaf is Healthy or Diseased and if diseased the exact type of diseased is shown.

5.2 System Output

The system predicts the input image given for classification into four different classes, namely Healthy leaf, Leaf galls, Anthracnose and Tuberculosis. Along with the class it also shows the confidence of prediction of the system.

Below are few output images with the type of diseases classified by the Crop disease Detection system. The below figures show different classes after classification by the system with confidence of predication.

(a) Classified as Healthy leaf

(b) Classified as Leaf Galls

(c) Classified as Tuberculosis

(d)Classified as Anthracnose

Fig 5.2 Different Output classes with confidence

The above Images represent the four different classes. In which Healthy leaf is classified with an average confidence of 97.6 percentage, Leaf Galls with 97.4 percentage, Tuberculosis with 99.7 percentage and similarly Anthracnose with confidence of 98.8 percentage. Hence the System gives an average overall accuracy of 94 percentage.

CONCLUSION

The Crop Disease Detection system is designed to increase yield in the agriculture sector by classification and predication of the exact type of disease based on the symptoms that appear on the leaf. The system uses a deep learning framework for identification and classification of the diseases. The system under consideration will detect the presence of disease and classify it from the healthy using Transfer learning on Resnet18 deep network. The model is further deployed on the hardware jetson nano to perform the real time disease classification. This system is more reliable, efficient and easy to use. It provides more accurate results than compared to traditional approaches. Recognizing the disease accurately and efficiently is mainly the purpose of the proposed approach. The experimental results indicate that the proposed approach is a valuable approach, which can significantly support an accurate detection of leaf diseases in a little computational effort. Alongside the supply of cultivation tools, the farmers also need access to accurate information that they can use for efficient crop management and there is no better way than providing them a service that they can use through the software.

FUTURE WORK

- To improve recognition rate of final classification process hybrid algorithms
- Mobile application can be developed which is handy and easy to use.
- An extension of this work will focus on automatically estimating the severity of the detected disease.
- As future enhancement of the project is to develop the open multimedia (Audio/Video) about the diseases and their solution automatically once the disease is detected

REFERENCES

[1] Akshai KP, J.Anitha. “Plant disease classification using deep learning”. 2021 3rd International Conference on Signal Processing and Communication (ICPSC), pp 1-4, May 2021. DOI: 10.1109/ICSPC51351.2021.9451696

[2] Ebrahim Hirani, Varun Magotra , Jainam Jain, Pramod Bide. “Plant Disease Detection Using Deep Learning”. 2021 6th International Conference for Convergence in Technology (I2CT) DOI: 10.1109/I2CT51068.2021.941791,IEEE 2021.

[3] Achyut Morbekar, Ashi Parihar, Rashmi Jadhav . “Crop Disease Detection Using YOLO”. 2020 International Conference for Emerging Technology .(INCET) Belgaum, India. IEEE 2020

[4] M.Akila and P.Deepan, "Detection and Classification of Plant Leaf Diseases by using Deep Learning Algorithm," International Journal of Engineering Research & Technology (IJERT), SSN: 2278-0181, 2018.

[5] K. Elangoran, S. Nalini, 2011 “Detection and classification of leaf diseases using K-means-based segmentation and neural-networks-based classification.” Inform. Technol. J., 10: 267-275. DOI: 10.3923/itj.2011.267.275.

[6] Sandesh Raut, Karthik Ingale, “Review on leaf disease detection using Image Processing techniques.”

[7] Sagar Patil, Anjali Chandavale, “A Survey on Methods of Plant Disease Detection”

[8] Mr. Sanjay Mirchandani, Mihir Pendse, Prathamesh Rane, Ashwini Vedula “Plant disease detection and classification using image processing and artificial neural networks.”

[9] Savita N. Ghaiwat, Parul Arora “Detection and Classification of Plant Leaf Diseases Using Image Processing Techniques:”

[10] Jayamala k Patil, Rajkumar “Advances in image processing for plant disease detection”