CHAPTER 1 OVERVIEW

1.1. Introduction

Over 80 percent of the population of Nepal is involved in agriculture, which constitutes 41 percent of GDP. Diseases are the main problems that threaten tomato cultivation these require careful diagnosis and timely handling to protect the crops form heavy loss. Problems growing tomatoes are often the result of weather conditions. This is something that is out of the gardener's control. Diseases in plants have been largely studied in the scientific area, mainly focusing on the biological characteristics of diseases. For instance, studies on potato and tomato show how susceptible a plant is to be affected by diseases. The problem of plant diseases is a worldwide issue also related to food security. Regardless of frontiers, media, or technology, the effects of diseases in plants cause significant losses to farmers. An earlier identification of disease is nowadays a challenging approach and needs to be treated with special attention.

In tomato plant disease can be found in various parts such stem, leaves and fruit Major diseases that affect tomato are: Tomato Diseases – Foliage (Early Blight, Gray Leaf Spot, and Late Blight etc) Tomato Diseases – Fruit (Anthracnose, Bacterial Speck, etc)

(a) (b) (c)

(d) (e) (f)

Figure 1.1: Leaf infected by common disease (a), (d) by early blight, (b), (e) by late blight and (c) (f) by septoria.

1.2 Problem Definition

In current scenario of Nepal, farmers are suffering from lack of knowledge to indentify the exact disease and its type especially in tomato farming .In addition to this; they are also unable to access not only the lab resources but also human expert in their territory.
It is very important to address problem technically and efficiently. This thesis specifically focus to solve these problem in tomato plant by building a system which takes input as leaf of tomato plant and classify the type of diseases based on input provided.

1.3 Objective

- To predict diseases namely early blight, late blight and septoria present in leaf of tomato
- To compare the accuracy of proposed model with other model

1.4 Scope of the work

The study work is focused on creating imputation model for accurately predicting whether leaf of tomato are infected by early blight, late blight and septoria disease or not which can be used in several areas especially in agriculture field of tomato farming.

CHAPTER 2 LITERATURE REVIEW

A number of approaches have turned to computer vision and machine learning techniques to create a fast method for plant diseases detection at the early onset of the symptom. Most of the studies presented in the literature of plant disease identification follow the steps shown in the Figure 2.1 [1]. As shown in Figure 2.1, the identification process starts by an image acquisition step where different digital devices are used to capture healthy and infected plant images. Then, further analysis is needed to edit the image and prepare it for later treatment, such as image enhancement, segmentation, color space conversion and filtering. In particular, image segmentation methods, like thresholding, are frequently used to detect boundaries in images.

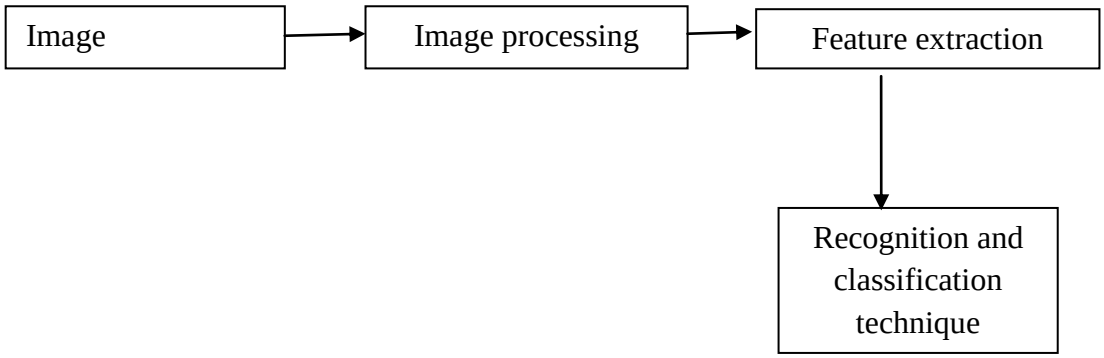

Figure 2.1: General steps applied to plant disease identification

Within the feature extraction step, features such as color, shape and texture are calculated from the image. Finally, the classification step is performed. Different classification algorithms are used in the literature such as neural network [2], and support vector machine [3]. In the following, we present a set of the state-of-the-art approaches following the general architecture in Figure 2.1.

In [4] applies an image processing methods for quantitatively detecting rust severity from multi-spectral images. A fast manual threshold-setting method was originally developed based on HSI (Hue Saturation Intensity) color model for segmenting infected areas from plant leaves. Two disease diagnostic parameters, ratio of infected area (RIA) and rust color index (RCI), were extracted and used as symptom indicators for quantifying rust severity.

In [5] proposes a method for disease identification, based on color transformations, color histograms and a pair wise-based classification system. Its performance was tested using large database containing images of symptoms belonging to 82 different biotic and abiotic stresses, affecting the leaves of 12 different plant species.

Even though different methods have achieved good classification results in identifying and recognizing some of the diseases, they suffer from some limitations. For example, segmentation is used in most methods as the first step in the leaf disease analysis. If the leaf image is captured with a black background, the segmentation is straightforward and no obstacles should be faced. However, when the background contains other leaves or plants, the segmentation may be questionable. Most of the methods will fail to effectively extract the leaf from its background which will lead to unreliable results. Also, some disease symptoms do not have well represented edges and they could gradually fade into healthy tissue. This may disturb solutions like color based methods and thresholding. Furthermore, a number of the methods rely on hand-crafted features such as color histograms, texture features, shape features and SIFT that requires expensive work and depends on expert knowledge. However, these methods do not generalize well and they are not effective when dealing with a large amount of data that could contain significant varieties [6].

Numerous procedures are currently in use for plant disease detection applying computer vision. One of them is disease detection by extracting color feature as authors in [7] have presented. In this paper YcbCr, HSI, and CIELB color models were used in the study; as a result, predictions were successfully done.

Subsequently, due to the recent advance in Machine Learning, the principle of CNN has been applied to plant diseases recognition in different crops, such as [8] using a CNN-based LeNet and image processing to recognize two leaf diseases out of healthy ones.

Another approach for cucumber leaf diseases, [9], used a three-layer CNN to train images containing two diseases out of healthy ones. To support the application of machine learning, [10] proposed to use a method called Color and Oriented FAST and Rotated BRIEF (ORB) to extract features and tree classifiers (Linear Support Vector Classifier (SVC), K-Nearest Neighbor, Extremely Randomized Trees) to recognize four types of diseases in cassava.

In [11], an image processing and statistical inference approach was introduced to identify three types of leaf diseases in wheat. In [12], the authors developed a method to discriminate good and bad condition images which contain seven types of diseases out of healthy ones in cucumber leaves. For that effect, they used an image-processing technique and a four-layer CNN, which showed an average of 82.3% accuracy under a 4-fold cross-validation strategy.

The pattern recognition system achieved an average accuracy of 85%. Islam et al. presented an approach that integrated image processing and machine learning to allow the diagnosis of diseases from leaf images. This automated method classifies diseases on potato plants from 'Plant Village', which is a publicly available plant image database. The segmentation approach and utilization of an SVM demonstrated disease classification in over 300 images, and obtained an average accuracy of 88% [13].

A handcrafted method is called so because of all the human knowledge implied in the development of the algorithm itself and the complex parameters that are included in the process. Some disadvantages of these methods are also the high computational cost and time consumption due to the complex preprocessing, feature extracting, and classifying. Some of the best-known handcrafted feature methods are the Histogram of Oriented Gradients (HOG) [14] and Scale-Invariant Feature Transform (SIFT) [15], which are usually combined with classifiers such as Support Vector Machines (SVM) [16].

CHAPTER 3 METHODOLOGY

3.1 Model development

To deal with the mentioned challenges, convolution neural network combined with recurrent neural network is used to classify and identify tomato leaves diseases. The general architecture of the proposed framework is illustrated in Figure 3.1. In the following, we present details about each component.

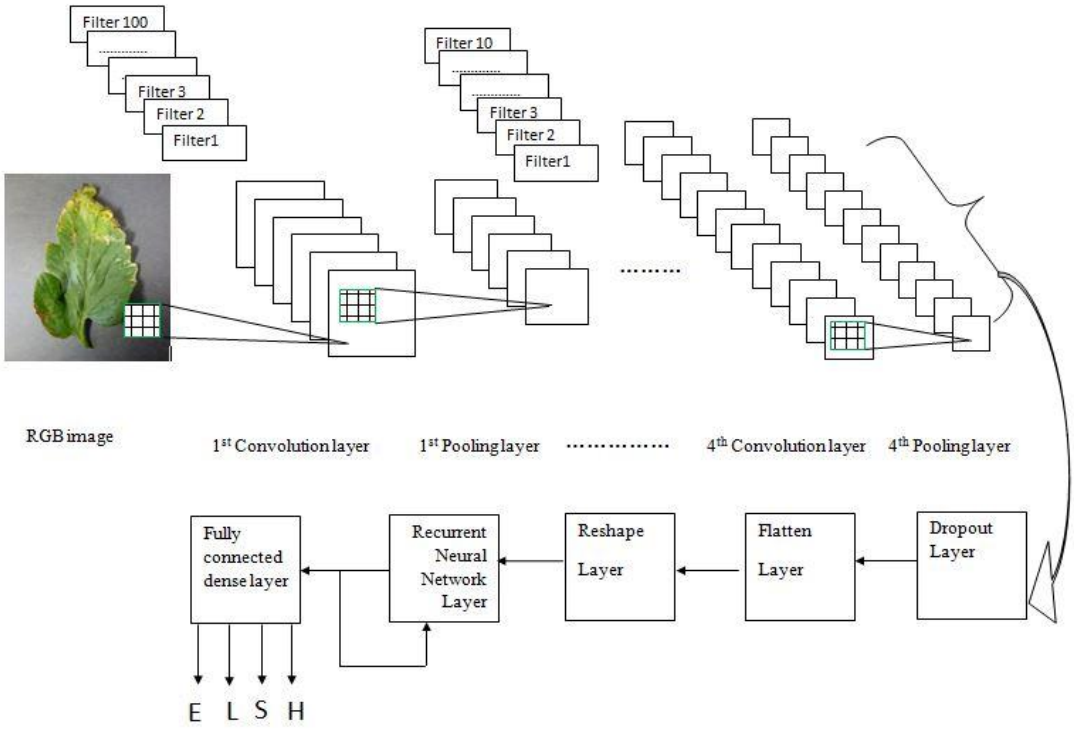

Figure 3.1: Proposed framework architecture

3.1.1 Dataset Preparation

Data provided is first being divided into **training set**, **validation set** and **testing sets**. Training sets are used to adjust weights and parameters of the model. Validation set are not used to adjust the parameters of the model, instead they are used to reduce over fitting problem. Testing set is used for evaluating the predictive power of the model.

3.1.2 Image processing

The dataset stored contains 2400 images of healthy and infected leaves. Each image has 3 channels which are red (R), green (G), and blue (B). Image in dataset are resized.

3.1.3 Convolution and recurrent neural network

Convolution neural networks (CNNs) are a kind of feeding forward neural network where every single node can be used to apply filters through overlapping regions. The processing occurs in an alternative fashion between convolution and sub-sampling layers followed by RNN and one fully connected layer such as standard multilayer perceptron (MLP). This architecture has various benefits compared to the standard Neural Networks.

The NNs have been successfully applied to features that have been extracted from other systems, which mean that the performance of NNs depends on matching relevant features that can be obtained. Another way to use NNs is to apply them directly to the raw pixel of the image. However, if the images have high dimensions, more parameters are needed because the hidden layer would be fully connected. To tackle this problem CNNs could be applied. The CNNs depend on sharing the weights, which reduces the numbers of parameters.

The convolution layers apply a local filter to the input image, which leads to a better classification there are correlation in the neighborhood pixels of the same image. In other words, the pixels of the input images can have some correlations with each other. For instance, the nose is always between the eyes and the mouth in face images. When we apply the filter to a subset of the image, we will extract some local features. By combining them subsequently, we will get the same format as the original image but with less dimensional image. These kinds of formats are not found in the fully connected layers.

3.1.3.1 Convolution Layers

The convolution layers can be mainly divided into two parts: Part 1 is a linear feature mapping that can be done by applying fixed size filters on the output of former layers.

$Y^l=W^l \otimes X^{l-1}$ …….....(3.1)

Where $\otimes$ denotes the convolution operator Part 2 involves convolution layers which is usually a nonlinear mapping as the ReLu function. The Rectified Linear Unit has become very popular in the last few years. It computes the function

f(x)=max(0,x)……..……..(3.2)

In other words, the activation is simply threshold at zero

3.1.3.2 Pooling Layers

Pooling layers usually receive their inputs from convolution layers. The pooling or the so called sub sampling layers is average or max operators over small squared areas. Based on the operator used, these layers are called average pooling or max pooling.
The average pooling can be defined as:

$$s_i = \frac{1}{n} \sum_{i \in R_j}^{n} h_j \text{ ..... } \text{.....(3.3)}$$

Where h is some pixel in the sub-region Rj from the features mapping, and n is number of features in the sub-region where sub sampling is required.
The max pooling can be defined as:

$$s_i = max_{i \in R_j} h_i \text{ .....(3.4)}$$

The goals of pooling layers are to reduce the feature mapping sizes and provide a connection to the local neighborhood of the convolution layer feature by doing their operations on a sequence of mapping features. Both the average and max pooling operators have some drawbacks. The average pooling pays attention to all elements in the pooling region even those that have ≤ zero value, which leads to a reduction in weight magnitudes. On the other hand, the max pooling can easily over fit the network.

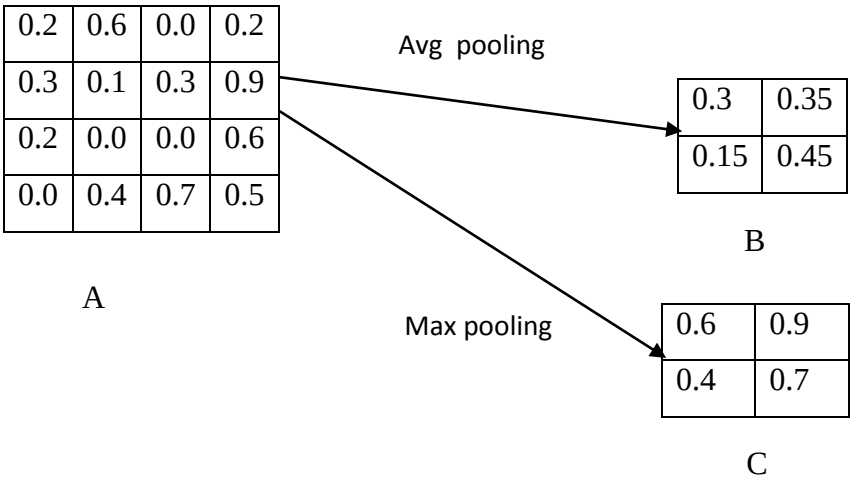

Figure 3.2: The pooling operations

A represents feature mapping from a previous layer, B represents the features that resulted from Average pooling, and C represents features that resulted from max pooling.

3.1.3.3 Dropout layer

Dropout layers have a very specific function in neural networks. The problem of over fitting, where after training, the weights of the network are so tuned to the training examples they are given that the network doesn't perform well when given new examples. The idea of dropout is simplistic in nature. This layer "drops out" a random set of activations in that layer by setting them to zero.

3.1.3.4 Flatten layer

Once the featured map is obtained, the next step is to flatten it. Flattening involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing. Flattening is the process of converting all the resultant 2 dimensional arrays into a single long continuous linear vector.

3.1.3.5 Recurrent Neural Networks

In a RNN, the information cycles through a loop. When it makes a decision, it takes into consideration the current input and also what it has learned from the inputs it received previously. A Recurrent Neural Network is able to remember exactly that, because of its internal memory. It produces output, copies that output and loops it back into the network.

During the training of RNN, as the information goes in loop again and again which results in very large updates to neural network model weights. This is due to the accumulation of error gradients during an update and hence, results in an unstable network. At an extreme, the values of weights can become so large as to overflow and result in NaN values. The drawback of RNN is over come by a new variant of the RNN model, called Long Short Term Memory. LSTM can solve this problem, because it uses gates to control the memorizing process.

Long Short-Term Memory (LSTM) networks are an extension for recurrent neural networks, which basically extends their memory. LSTM's enable RNN's to remember their inputs over a long period of time.

With RNN, the connections are no longer purely feed-forward. As its name implies, there is now a recurrent connection that connects the output of a RNN neuron back to itself. Figure 3.3 shows a single RNN neuron.

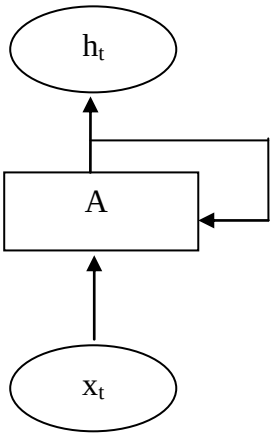

Figure 3.3: Basic Structure of Recurrent Neural Network

In this picture, the input, x_t is the input at time *t*. As in the feed-forward case, we feed the input into our neuron (block A), it does some computation, and we get the output h_t. However note an additional recurrent connection feeding the same output h_t back into A.

What happens with this neuron at the next time step? Well, we will get another input, x_t+1 and will feed that into our neuron (Block A), however recall our recurrent connection from the previous time step, h_t, this is also part of our input. In a simple function mapping, a RNN's computation will be, following the diagram notation above:

$h_t=f(x_t,h_t-1)$……(3.5)

This means that for RNN, for each neuron, there're two weights, a feed-forward weight (just like we would have in a MLP) and a recurrent weight.

Long Short Term Memory Networks

LSTMs are special kind of RNNs with capability of handling Long-Term dependencies. LSTMs also provide solution to Vanishing/Exploding Gradient problem. A simple LSTM cell representation is shown in figure 3.4.

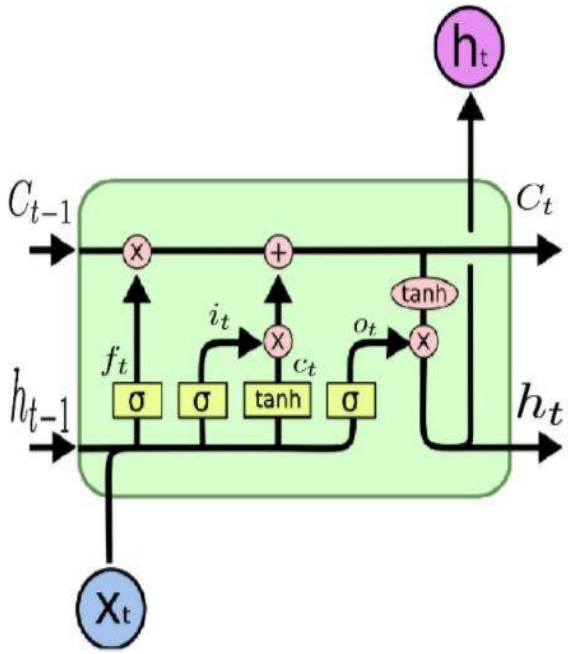

Figure 3.4: LSTM cell visual representation [17]

Forget Gate: After getting the output of **previous state**, **h(t-1)**, Forget gate helps us to take decisions about what must be removed from h(t-1) state and thus keeping only relevant stuff. It is surrounded by a sigmoid function which helps to crush the input between [0, 1].It is represented as in figure 3.5.

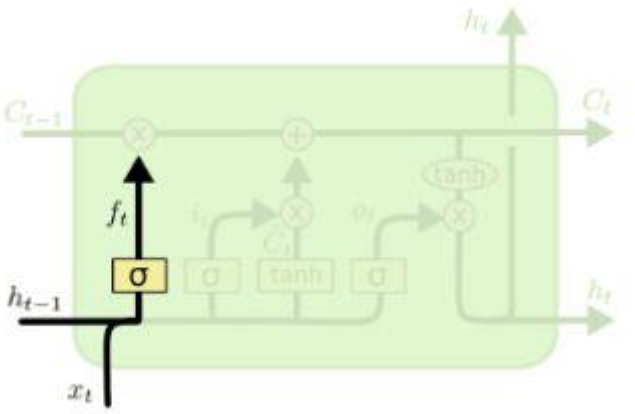

Figure 3.5: Forget gate visual representation [17]

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f) \quad \text{.....(3.6)}$$

We multiply forget gate with previous cell state to forget the unnecessary stuff from previous state which is not needed anymore.

Input Gate: In the input gate, we decide to add new stuff from the present input to our present cell state scaled by how much we wish to add them.

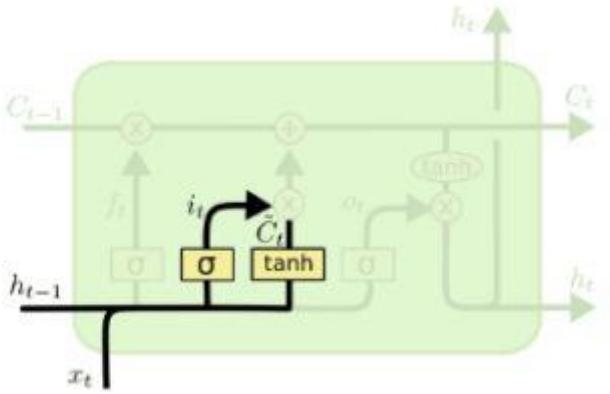

Figure 3.6: Input gate visual representation [17]

$$i_t = \sigma(\, W_i . [h_{t-1}, x_t] + b_i \,) \quad \text{.....(3.7)}$$

$$\dot{C}_t = \tanh(W_C . [h_{t-1}, x_t] + b_C) \quad \text{.....(3.8)}$$

In figure 3.6, sigmoid layer decides which values to be updated and tanh layer creates a vector for new candidates to added to present cell state**.**

To calculate the present cell state, we add the output of ((input_gate*gate_gate) and forget gate) as shown below:

$$C_{t=} f_t + C_{t-1} + i_t * \dot{C}_t \quad \text{.....(3.9)}$$

Output Gate: Finally we'll decide what to output from our cell state which will be done by our sigmoid function.

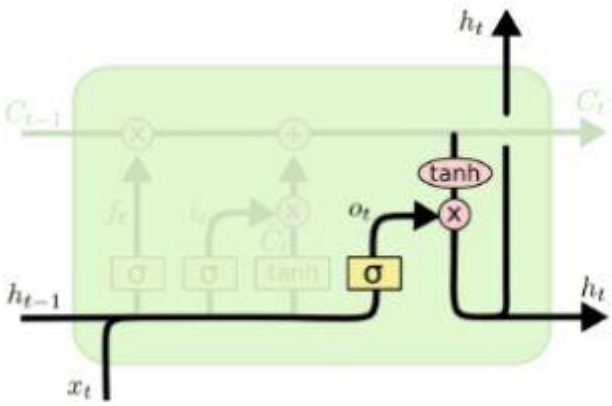

Figure 3.7: Output gate visual representation [17]

$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$……...(3.10)

$h_t = o_t * \tanh(C_t)$……...(3.11)

As shown in Figure 3.1 system diagram is composed of four main parts which are convolution, pooling, recurrent and fully connected layers. The convolution and pooling layers act as feature extractors from the input images while RNN acts as memory and the fully connected layer acts as a classifier. The essential purpose of convolution is to extract features automatically from each input image. The dimensionality of these features is then reduced by the pooling layer. At the end of the model, the fully connected layer with a softmax activation function makes use of the learned high-level features to classify the input images into predefined classes.

3.1.3.6 Fully Connected Layers

After alternating between the convolution and the sub sampling operations, and performing dropout, flatten and reshape operation a single long continuous linear vector is passed to RNN where it produces output to fully connected layer, copies that output and loops it back into the network.

The fully connected layer is similar to the hidden layer in ANNs but in this case it's fully connected. The output layer is where we get the predicted classes. The information is

passed through the network and the error of prediction is calculated. The error is then back-propagated through the system to improve the prediction.

3.1.3.7 Back propagation Algorithm

There are two passes in the back propagation algorithm, the forward and backward pass.

Forward Propagation

To simplify illustration, we assume our CNN has only one convolution layer, one sub sampling layer, one recurrent neural network layer and one fully connected layer.

For convolution layers, suppose we have an image x that has a size of $m \times m$ and a weight kernel w that has a size of k. So we shall have an output that has a size of $(m-k+1) \times (m-k+1)$ after we convolve the input image with the kernel. The convolution process is a dot product between the weight and part size of the input that has a size equal to the weight; after that, we sum over all the dot product results.

$$y_{ij}^{l} = \sum_{a=1}^{m} \sum_{b=1}^{m} w_{ab}^{l} \; x_{i+a,j+b}^{l-1} \quad \text{.....(3.12)}$$

where l denote the current layer and i, j defines the location of the next pixel in the output of the l th layer.

Every convolution layer has a normalization part defined as:

$$x_{ij}^{l} = f(y_{ij}^{l} + b^{l}) \quad \text{..... (3.13)}$$

where $f(\cdot)$ is the normalization function and commonly chosen to be the logistic (sigmoid) function and b^{l} is the bias.

The sub sampling layer: As noted in Section (3.1.3.2), there are two types of these layers. Neither of them have weights nor a normalization part. The output's size from this layer will drop to half if we have a kernel size of (2 x 2).

Backward propagation

The back-propagation process starts from the end layer to the first layer. This process would be similar to Neural Networks on the fully connected layers.

Back propagation for the sub sampling layer: The pooling layers need not to have any trainable parameters to be updated. Those kinds of layers can only serve to reduce the size of the features mapping, so, in the backward pass, there is no derivative operation required. In the forward pass, we do sub sampling over a square area that is reduced to a single value after the operation. In the backward pass, it is required to return a single value from the error to same size of the squared area. Let's call it dissampling. The dissampling operation depends on the kind of subsampling required. In the case of average pooling, the errors that computed from the layer before the pooling layer distribute on the square area equally. In the max pooling case, the error forwards directly to the place where the feature in the max pooling originated from and the remaining spaces are filled by zeros.

The back propagation in the convolution layers: If we know what errors occurred in the layer before the convolution layers, say E, then we can find the error in the convolution layer. Note that we have a square kernel that has a size of k X k. Hence, we need to perform the chain rule and consequently find the sum over all the regions.

$$\frac{\partial E^l}{\partial w_{ij}^l} = \sum_{i=1}^{m-k} \sum_{j=1}^{m-k} \frac{\partial E^{l+1}}{\partial y^l} \frac{\partial y^l}{\partial w_{ij}^l} \quad \text{.....(3.14)}$$

From equation (3.12) $\frac{\partial Y^l}{\partial W_{ij}^l} = \mathrm{x}_{\mathrm{ij}}^{\mathrm{l}-1}$ and we can get $\frac{\partial \mathrm{E}^{\mathrm{l}+1}}{\partial \mathrm{Y}^{\mathrm{l}}}$ by applying the chain rule again;

$$\frac{\partial \mathrm{E}^{\mathrm{l}+1}}{\partial \mathrm{y}^{\mathrm{l}}} = \frac{\partial \mathrm{E}^{\mathrm{l}+1}}{\partial \mathrm{x}_{\mathrm{ij}}^{\mathrm{l}}} \frac{\partial \mathrm{x}_{\mathrm{ij}}^{\mathrm{l}}}{\partial \mathrm{y}_{\mathrm{ij}}^{\mathrm{l}}} \quad \text{.....(3.15)}$$

Where $\frac{\partial x}{\partial y}$ from equation (3.12) equals the derivative of the activation functions; by putting all the terms together, we get

$$\frac{\partial E^l}{\partial w_{ij}^l} = \sum_{i=1}^{m-k} \sum_{j=1}^{m-k} \frac{\partial E^l}{\partial x_{ij}^l} x_{ij}^{l-1} f'(y_{ij}^l) \quad \text{.....(3.16)}$$

After that we update the weights accordingly using equation

$$w(t) = w(t-1) - \eta \frac{\partial E}{\partial w} \quad \text{.....(3.17)}$$

Algorithm

Step1: Initialize all filters and parameters / weights with random values

Step2: The network takes a training image as input, goes through the forward propagation step (convolution, and pooling operations along with RNN and forward propagation in the Fully Connected layer) and finds the output probabilities for each class.

Step 3: Calculate the loss for batch

$$-\sum_{c=1}^{M} y_{o,c} log(p_{o,c}) \quad \text{.....(3.18)}$$

Step 4: Use Back propagation to calculate the gradients of the error with respect to all weights in the network and use gradient descent to update all filter values / weights and parameter values to minimize the output error.

Step 5: Repeat steps 2-4 with all images in the training set.

The step1 to step 5 trains the network – this essentially means that all the weights and parameters of the network have been optimized using Adaptive moment estimation algorithm to correctly classify images from the training set.

When a new (unseen) image is input into the network, the network would go through the forward propagation step and output a probability for each class (for a new image, the output probabilities are calculated using the weights which have been optimized to correctly classify all the previous training examples).

3.2 Tools and technology

The following tools will be used in the research work:

- Python
- Visual studio code editor
- Deep learning framework(Keras and Tensorflow)

3.3 Data Collection

Collected from www.PlantVillage.org and www.narc.gov.np. Dataset consists of four thousand images of size 100X100. 70 % images are used for Training while 30% are Test image. There are 1000 images per class. Different classes of Dataset are:

Table 3.1: Classes of datasets and their corresponding label

Label	Class
0	BacterialSpot(Early blight)
1	septoriaLeaf Spot
2	Late blight
3	Healthy

CHAPTER 4 RESULT ANALYSIS AND DISCUSSION

4.1 Overview of task

4.1.1 Deep Learning Framework Installation and CPU Configuration

A very effective deep learning framework 'Tensor Flow' is installed and is configured to run on Core i5.TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them. A deep learning framework 'Keras' is used for efficient implementation.

4.1.2 Network Architectures

Architecture is designed with following layers' configuration:

i. INPUT Layer accepting input of 100X100 image with three channel color

ii. First CONV Layer with 100 3x3 filters. With RELU activation function.

iii. MAXPOOL Layer with size 2x2.

iv. Second CONV Layer with 50 3x3 filters. With RELU activation function.

v. Second MAXPOOL Layer with size 2x2

vi. Third CONV Layer with 25 3x3 filters. With RELU activation function.

vii. Third MAXPOOL Layer with size 2x2

viii. Dropout Layer with 25% dropout.

ix. Fourth CONV Layer with 10 3x3 filters. With RELU activation function.

x. Fourth MAXPOOL Layer with size 2x2

xi. Dropout Layer with 25% dropout.

xii) Flatten layer

xiii) Reshape layer which reshapes into 2X80

xiv. Recurrent layer having units 30

xv. . FULL Connection Layer with 4 units with softmax activation function

4.1.3 Experiment Results

Figure 4.1: Sample image

When figure 4.1 sample image is passed through 1st convolution layer its output is shown in figure 4.2

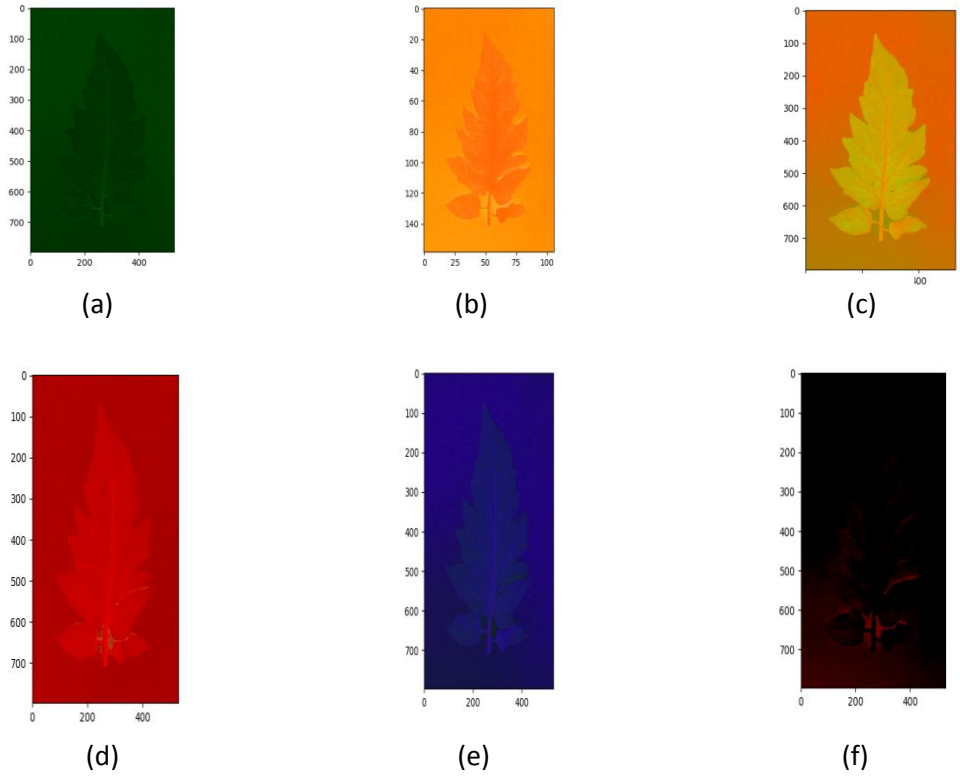

Figure 4.2: (a), (b), (c), (d), (e) and (f) are outputs from first convolution layer without activation and max pooling operation (Filter size 3X3)

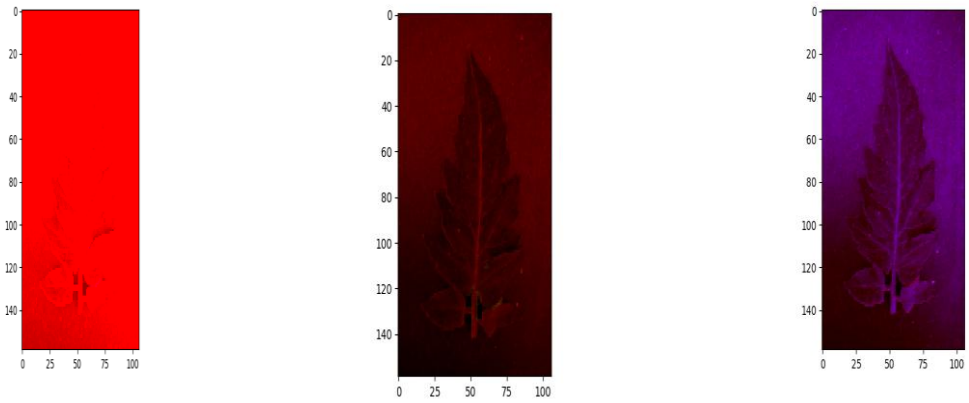

Figure 4.3 Outputs from first convolution layer with relu activation

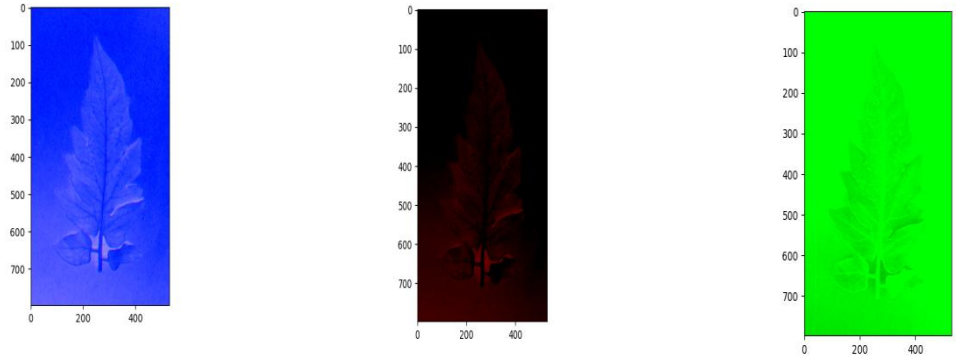

Figure 4.4 Outputs from first convolution layer with relu activation and max pooling operation

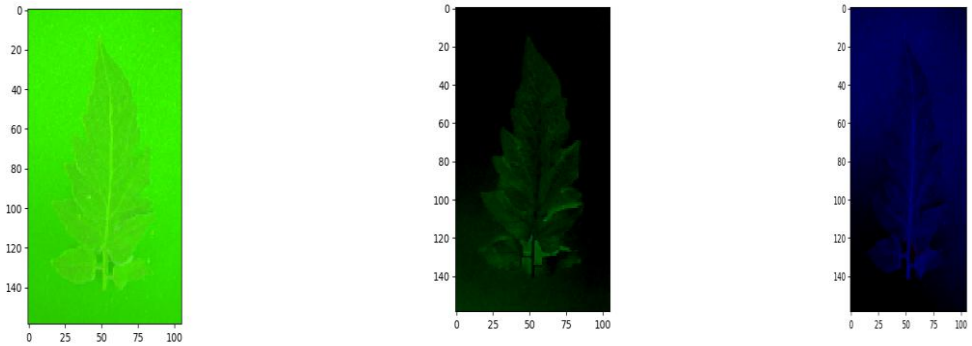

Figure 4.5 Outputs from first convolution layer without activation and max pooling operation (Filter size 5X5)

As we can see in figure 4.5, we lost some of the detail because the kernel was big (5X5) compared in figure 4.2 where kernel size is 3X3.

Output of first layer acts as input for second layer. Output from 2nd layer is shown in figure 4.7.

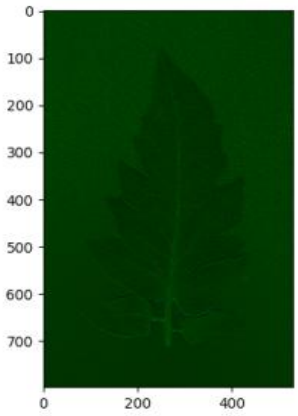

Figure 4.6 Sample input for Second Convolution layer

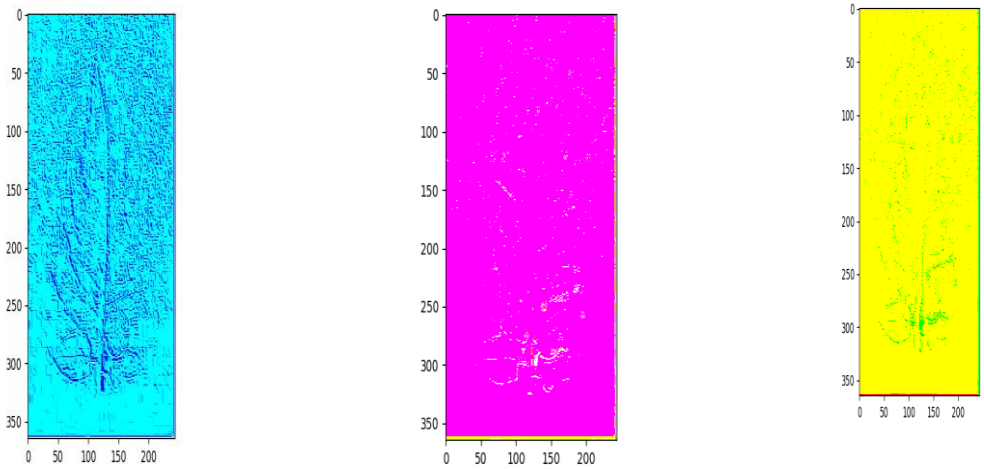

Figure 4.7**:** Output from Second convolution layer (Filter size 3X3)

Figure 4.2, 4.3, 4.5 and 4.7 shows that convolution neural network are powerful due to their ability to extract the core features of an image and use these features to indentify images that contain features like them. Even with our two layers CNN we can start to see the network is paying a lot attention to different regions.

4.1.4 Result Analysis

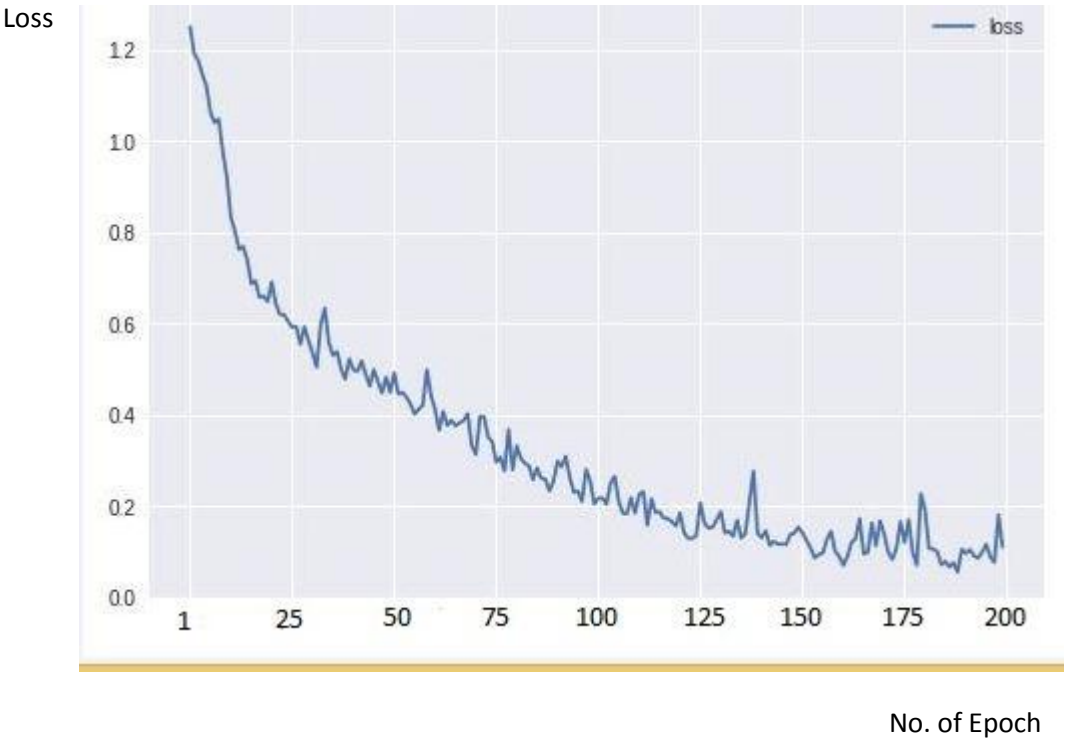

Figure 4.8: Loss when number of epochs is 200

The figure 4.8 shows loss when number of epochs is 200 which is equal to 0.1934 and achieved 92.16 % of accuracy.

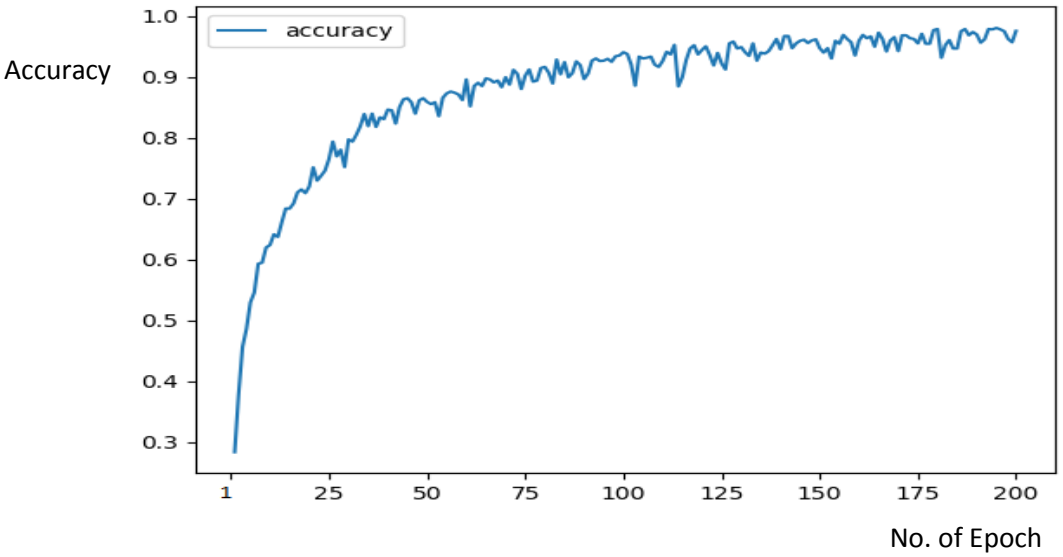

Figure 4.9: Accuracy when number of epochs is 200

The figure 4.9 shows accuracy when number of epochs is 200 which is equal to 92.16%.

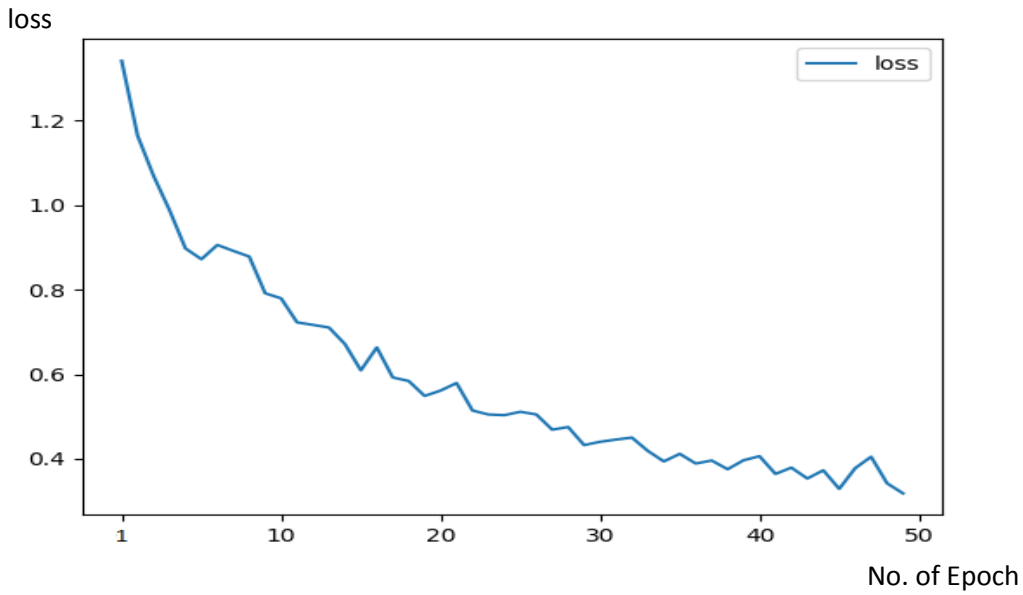

Figure 4.10: Loss when number of epochs is 50(training set 70%)

The figure 4.10 shows loss during training period when number of epochs is 50 which is equal to 0.3637 and achieved 85.75 % of accuracy. In this case percentage of training set equals to 80 and testing set equals to 20.

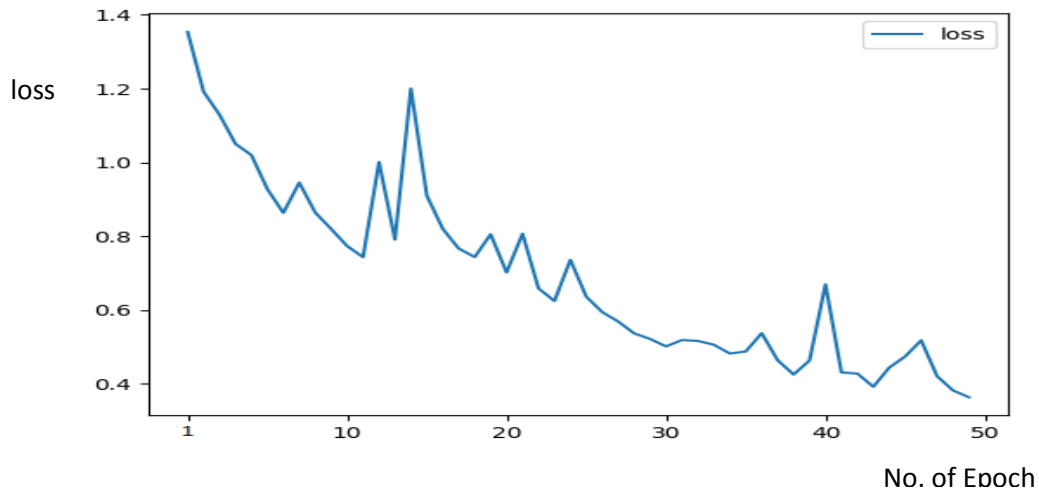

Figure 4.11: Loss when number of epochs is 50(training set 80%)

The figure 4.11 shows loss during training period when number of epochs is 50 which is equal to 0.44 and achieved 82.46 % of accuracy. In this case percentage of training set equals to 70 and testing set equals to 30.

Table 4.1 Performance comparison by assigning different values of hyper parameter

No_ of_e poch	L_rat e	Train set	Validti on_set	K_size	Pool size	Act functio n	Dropou t	Loss	accu racy
20	0.001	80%	20%	3X3	2X2	ReLU	25%	0.59	0.73
20	0.001	70%	30%	3X3	2X2	ReLU	25%	0.44	0.76
50	0.001	80%	20%	3X3	2X2	ReLU	25%	0.3637	0.85
50	0.001	70%	30%	3X3	2X2	ReLU	25%	0.44	0.82
100	0.001	70%	30%	3X3	2X2	ReLU	25%	0.2822	0.88
200	0.001	70%	30%	3X3	2X2	ReLU	25	0.1934	0.92

Table 4.1 shows loss and accuracy of network. Result shows that when number of epoch's increases and also number of dataset increases the performance of the system increase.

Further 40 standard images were selected for cross validating the model and the results were as follows:

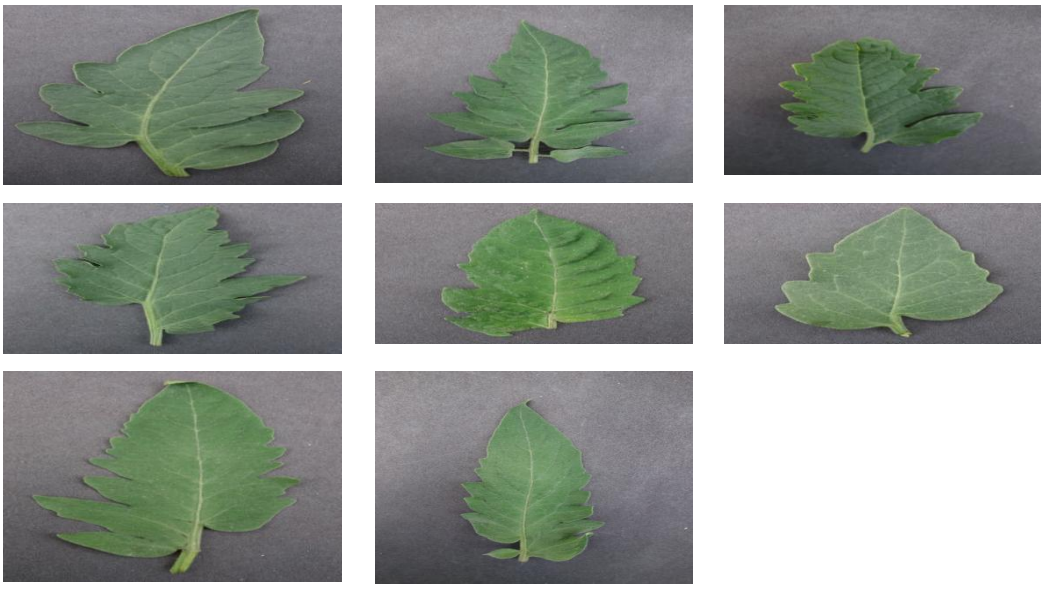

Figure 4.12: Healthy image

When figure 4.12 were tested against the model. All images are healthy. Healthy images are labeled as 3.The predicted result is as below:

```
feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was
 not compiled to use: AVX2
./test_folder2/h1.png [0.0142534  0.05261184 0.09590612 0.8372286 ]
./test_folder2/h2.png [0.01262534 0.07264144 0.13102448 0.7837087 ]
./test_folder2/h3.png [0.02009131 0.2033162  0.40695494 0.3696375 ]
./test_folder2/h4.png [0.01728955 0.08057123 0.23173507 0.67040414]
./test_folder2/h5.png [0.06084548 0.59234023 0.33728662 0.00952767]
./test_folder2/h6.png [0.1921529  0.6397846  0.16097933 0.00708321]
./test_folder2/h7.png [0.00636562 0.03537317 0.03787912 0.9203821 ]
./test_folder2/h8.png [0.00825793 0.0452459  0.04597093 0.9005253 ]
```

Figure 4.13: Healthy leaf tested against built model

Healthy leaf is labeled as 3. Figure 4.13 shows out of 8 images from validating set 5 images are predicted correctly.

Figure 4.14: Leaf infected by Septoria disease.

When figure 4.14 were tested against the model. All images are infected by Septoria disease. Leaf infected by Septoria images are labeled as 1.The predicted result is as below:

```
./test_folder1/sep1.jpg [0.6958578  0.2527391  0.04577209 0.005631  ]
./test_folder1/sep2.jpg [0.45351055 0.47937202 0.05741365 0.00970381]
./test_folder1/sep3.jpg [0.63234603 0.31320852 0.04569698 0.00874839]
./test_folder1/sep4.jpg [0.20257203 0.6499791  0.1356541  0.01179468]
./test_folder1/sep5.jpg [0.23068309 0.6813952  0.08184506 0.00607672]
./test_folder1/sep6.jpg [0.38518262 0.50873387 0.09386689 0.01221658]
./test_folder1/sep7.jpg [0.24496116 0.6555998  0.08898157 0.0104575 ]
./test_folder1/sep8.jpg [0.2663591  0.5484448  0.17400143 0.01119462]
```

Figure 4.15: Septoria Leaf spot tested against model

Septoria leaf spot is labeled as 1. Figure 4.15 shows out of 8 images from validating set 6 images are predicted correctly.

Figure 4.16 Leaf infected by late blight

When figure 4.16 were tested against the model. All images are infected by late blight disease. Leaf infected by late blight images are labeled as 2.The predicted result is as below:

```
./test_folder3/l1.png [0.00147258 0.6249863  0.37145534 0.00208578]
./test_folder3/l2.png [0.00074978 0.39216095 0.60589814 0.00119108]
./test_folder3/l3.png [5.4175919e-04 2.6755100e-01 7.3007917e-01 1.8280775e-03]
./test_folder3/l4.png [0.0008262  0.4053648  0.5924031  0.00140589]
./test_folder3/l5.png [6.0110429e-04 2.7847162e-01 7.1875459e-01 2.1726780e-03]
./test_folder3/l6.png [0.00083374 0.45788664 0.5396084  0.0016712 ]
./test_folder3/l7.png [5.4175919e-04 2.6755100e-01 7.3007917e-01 1.8280775e-03]
./test_folder3/l8.png [0.00183828 0.84746605 0.14957327 0.00112242]
```

Figure 4.17: Late blight tested against model

Late blight is labeled as 2. Figure 4.17 shows out of 8 images from validating set 6 images are predicted correctly

Figure 4.18: Two image of each class

When figure 4.18 were tested against the model. All images are infected by late blight disease. Leaf infected by late blight images are labeled as 2.The predicted result is as below:

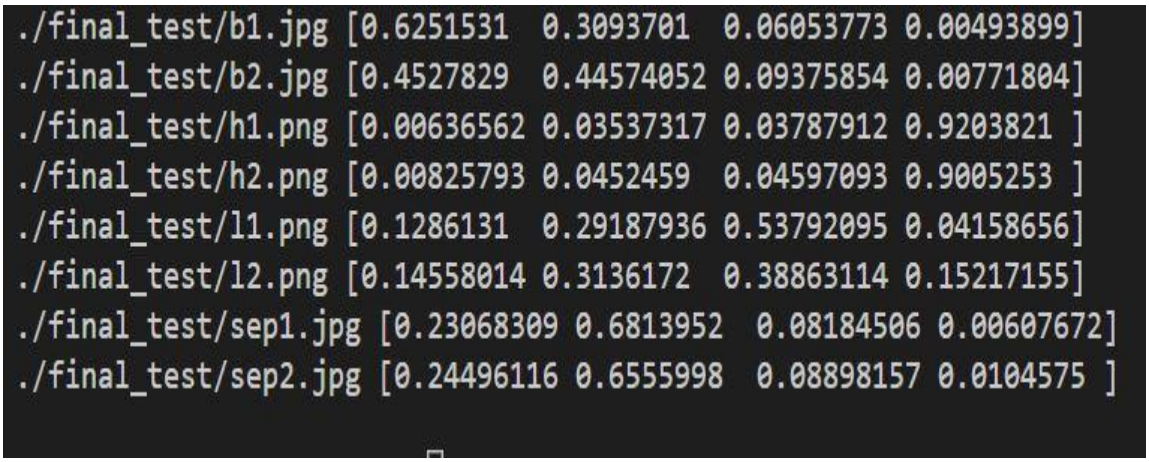

Figure 4.19: Two images from each class are tested against model

As in our model b1, b2 represent bacterial spot labeled as 0, sep1, sep2 represent sepotoria labeled as 1, l1, l2 represent late blight labeled as 2 and h1 and h2 represent healthy image labeled as 3. Figure 4.19 shows out of 8 images from validating set all are predicted correctly.

Table 4.2: Number of Correct and wrong prediction

Input	**No**	**Correct Predicted**	**Wrong Predict**
Healthy_image	8	5	3
Septoria leaf	8	6	2
Late blight	8	6	2
Early blight	8	8	0
2 images from each class	8	8	0
Result	40	33	8

Data present in Table 4.2 is taken from figure 4.13, 4.15, 4.17and 4.19.

4.1.5 Confusion matrix Analysis

Table 4.3 Confusion matrix of model

	Predicated				
		Early blight(bacterial spot)	**Septoria leaf spot**	**Late Blight**	**Healthy**
Actual	Early blight(bacterial spot)	10	2	0	0
	Septoria leaf spot	0	8	2	2
	Late Blight	0	0	8	1
	Healthy	0	0	0	7

Table 4.3 illustrates confusion matrix for these report. it consists of four classes (Early blight, bacterial spot, Septoria leaf spot, Late Blight, Healthy) having 10 images of each class and the diagonal elements represents true positive value.

Table 4.4, 4.5, 4.6 and 4.7 were derived from table 4.3 which show binary confusion matrix of each class.

Table 4.4: Confusion matrix for early blight

		Actual class	
		Early blight	Non- Early blight
Predicted class	Early blight	10 TP	0 FP
	Non-Earlyblight	2FN	28 TN

Table 4.5: Confusion matrix for septoria leaf spot

		Actual class	
		septoria leaf spot	Non- septoria leaf spot
Predicted class	septoria leaf spot	8 TP	2 FP
	Non-septoria leaf spot	4FN	26 TN

Table 4.6: Confusion matrix for late blight

		Actual class	
		Late blight	Non- late blight
Predicted class	Late blight	8 TP	2 FP
	Non- Late blight	1FN	29 TN

Table 4.7: Confusion matrix for healthy

		Actual class	
		Healthy	Non- Healthy
Predicted class	Healthy	7 TP	3 FP
	Non-Healthy	0FN	30 TN

Overall accuracy=(10+8+8+7)/40=0.85

Precison Calculation(TP/TP+FP)

Precison of healthy leaf=7/(7+3)=0.7

Precison of late blight leaf=8/(8+2)=0.8

Precison of septoria leaf spot=8/(8+2)=0.8

Precison of Early blight=10/(10+0)=1

Avg precison=(0.7+0.8+0.8+1)/4=0.825

Recall or Sensitivity Calculation (TP/TP+FN)

Sensitivity of healthy leaf)=7/(7+0)=1

Sensitivityof late blight leaf=8/(8+1)=0.89

Sensitivity of septoria leaf spot=8/(8+4)=0.67

Sensitivity of Early blight=10/(10+2)=0.84

Avg Sensitivity=(1+0.89+0.67+0.84)/4=0.85

Specificity Calculation (TN/TN+FP)

Specificity of healthy leaf=30/(30+3)=0.91

Specificity of late blight leaf=29/(29+2)=0.93

Specificity of septoria leaf spot=26/(26+2)=0.92

Specificity of Early blight=28/(28+0)=1

Avg Specificity=(0.91+0.93+0.92+1)/4=0.94

Fmeasure(healthy)=2*precision*recall/precision+recall=2*0.7*1/1+0.7=0.82

Fmeasur(Early blight)=2*1*0.84/(1+0.84)=0.913

Fmeasure(seproria)=2*0.8*0.67/(0.8+0.67)=0.73

Fmeasure(late blight)=2*0.8*0.89/0.8+0.89=0.842

Fmeasure(Avg)=0.826

Table 4.8: Comparison with other model

Model name	Accuracy	loss
Support vector machine[3]	0.82	0.36
CNN[8]	0.72	0.42
CNN with RNN	0.92	0.1934

4.1.6 Discussion

It is found that the developed model is pretty much able to learn feature of input image and predict disease present in leaf of tomato with accuracy 92.16%. This research presents the study on plant diseases detection using artificial neural network. The optimum result shows that proficiency of combined CNN with RNN in recognition of tomato plant disease using infected leaf's image.

As expected the accuracy of CNN model was the least due to not an integer problem. So, RNN was added to overcome this problem. In addition, SVM layer introduced to CNN for further comparison. The accuracy and losses of these models are illustrated in table 4.8.

From table 4.8 it is clear that CNN has the least accuracy and adding RNN layer to CNN layer increased the models accuracy, which was greater than adding SVM layer to CNN.

CHAPTER 5 CONCLUSION AND RECOMMENDATION

5.1. Conclusion

Agriculture suffers from a severe problem, plant diseases, which reduce the production and quality of yield. This thesis work focus to detect disease present in leaf of tomato plant and involves collecting four class of tomato leaf, i.e., healthy leaf, leaf infected by early blight, late blight and septoria. Work was carried out to investigate the use of computer vision for classifying tomato leaf disease.

This thesis presented an approach based on convolution neural networks along with recurrent neural network to identify and classify tomato leaf diseases. The proposed model can serve as a decision support tool to help farmers to identify the disease in the tomato leaf. Hence, the farmer can take a picture of the leaf with the symptoms and then the system will identify the type of the disease. Main contribution is to apply deep neural networks along with recurrent neural network to detect common diseases such early blight, late blight and septoria.

5.2 Limitation and Future Enhancement

As a limitation; this system is only capable of detecting three classes of diseases and healthy plant. In order to detect other class of diseases data has to be trained on current model. Algorithm will use transfer learning method to classify other class of diseases. The main challenge while developing object detection model on machine learning was to collect large number of train images with different shapes, sizes, with different background, light intensity, orientation and aspect ratio.

As per the recommendation; the further study can be detect all types of plant diseases, not only detection but also suggesting remedies for diseases. Finally, integrated with IOT server to implement system on rural and remote area.

To overcome problem of convolution neural network another class of deep neural network named capsule network can be use to get better result.

REFERENCES

[1] Al-Hiary, H.; Bani-Ahmad, S.; Reyalat, M.; Braik, M.; ALRahamneh, Z.: Fast and accurate detection and classification of plant diseases. International Journal of Computer Applications, 17(1):0975–8887, 2016.

[2] Sannakki, Sanjeev S; Rajpurohit, Vijay S; Nargund, V B; Kulkarni, Pallavi: Diagnosis and Classification of Grape Leaf Diseases using Neural Networks. In: Fourth International Conference on Communications and Networking Technologies. 2013.

[3] Elangovan, K; Nalini, S: Plant Disease Classification Using Image Segmentation and SVM Technique. International Journal of Computational Intelligence Research, vol 13, no.7,pp.1821-1828, 2017.

[4] Cui, Di; Zhang, Qin; Li, Minzan; Hartman, Glen L.; Zhao, Youfu: Image processing methods for quantitatively detecting soybean rust from multispectral images. Biosystems Engineering,107(3):186–193, 2010.

[5] Barbedo, Jayme Garcia Arnal; Koenigkan, Luciano Vieira; Santos, Thiago Teixeira: Identifying multiple plant diseases using digital image processing. Biosystems Engineering,147:104–116, 2016.

[6] Schor, Noa; Bechar, Avital; Ignat, Timea; Dombrovsky, Aviv; Elad, Yigal; Berman, Sigal: Robotic Disease Detection in Greenhouses: Combined Detection of Powdery Mildew and Tomato Spotted Wilt Virus. IEEE ROBOTICS AND AUTOMATION LETTERS,1(1):354–360, 2016.

[7] P. Chaudhary, A.K. Chaudhari, A.N. Cheeran, and S. Godara: Color transform based approach for disease spot detection on plant leaf. International Jounal of Computer Science and Telecommunications, vol 3,no. 6,pp 65-69,2012

[8] Amara J., Bouaziz B., Algergawy A. Lecture Notes in Informatics (LNI) Gesellschaft für Informatik; Bonn, Germany: 2017. A Deep Learning-based Approach for Banana Leaf Diseases Classification.

[9] Kawasaki Y., Uga H., Kagiwada S., Iyatomi H. Basic Study of Automated Diagnosis of Viral Plant Diseases Using Convolutional Neural Networks. In: Bebis G., editor. Advances in Visual Computing, Proceedings of the 11th International Symposium,

ISVC 2015, Las Vegas, NV, USA, 14–16 December 2015. Volume 9475. Springer; Cham, Switzerland: 2015. pp. 638–645. Lecture Notes in Computer Science..

[10] Owomugisha, G.; Mwebaze, E. Machine Learning for Plant Disease Incidence and Severity Measurementsfrom Leaf Images. In Proceedings of the 2016 15th IEEE International Conference on Machine Learning andApplications (ICMLA), Anaheim, CA, USA, 18–20 December 2016.

[11] Johannes A., Picon A., Alvarez-Gila A., Echazarra J., Rodriguez-Vaamonde S., Diez-Navajas A., Ortiz-Barredo A. Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case. Comput.Electron. Agric. 2017;138:200–209. doi: 10.1016.compag.2017.04.013.

[12] Fujita E., Kawasaki Y., Uga H., Kagiwada S., Iyatomi H. Basic investigation on a robust and practical plant diagnostic system; Proceedings of the 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA); Anaheim, CA, USA. 18–20 December 2016.

[13] Islam, A. Dinh, K. Wahid and P. Bhowmik, "Detection of potato diseases using image segmentation and multiclass support vector machine," Conference on Electrical and Computer Engineering, Canada, 30 April–3 May 2017.

[14] Dalal, N.; Trigs, B. Histogram of Oriented Gradients for Human Detection. In Proceedings of the IEEEComputer Society Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–25June 2005.

[15] Lowe, D. Distinctive Image Features from Scale-Invariant Keypoints. Int. J. Comput. Vis. 2004, 60, 91–110.

[16] Cortes, C.; Vapnik, V. Support Vector Networks. Mach. Learn. 1995, 20, 293–297.

[17] Manik Soni- https:hackermoon.com/Understanding- architecture-of-lstm-cell-from scratch