Disease Detection and Classification in Agricultural Plants Using Convolutional Neural Networks – A Visual Understanding

Mercelin Francis
Dept. of Computer Science & Engineering
Thiagarajar College of Engineering
Madurai, India
mercelinf@gmail.com

C. Deisy
Dept. of Information Technology
Thiagarajar College of Engineering
Madurai, India
cdcse@tce.edu

***Abstract*— Convolutional Neural Network is the base of all deep learning models. Therefore a Convolutional Neural Network model is created and developed to perform plant disease detection and classification using apple and tomato leaf images of healthy and diseased plants. The model consists of four convolutional layers each followed by pooling layers. Two fully connected dense layers and sigmoid function is used to detect the probability of presence of disease or not. Training of the model was done on apple and tomato leaf image dataset containing 3663 images achieving an accuracy of 87%. The overfitting problem is identified and removed setting the dropout value to 0.2. As the model allows parallel processing, it is also run on GPU Tesla to evaluate its speed of performance and accuracy. Hence the paper provides an insight of creativeness to the researchers to develop an integrated plant disease identification system that gives successful results in realtime.**

Keywords*— *Deep Learning; Convolutional Neural Networks; Plant Disease Classificatiopn; Image Processing; Machine Learning; Information and Communication Technology (ICT)

I. INTRODUCTION

Disease found in agricultural crops is a major threat that affects the economic impact, its yield and production, if left unattended on-time. The crop losses can be minimized by applying pesticides or its equivalent to combat the effect of specific pathogens if diseases are correctly diagnosed and identified early. Various on-field observation (viz. analysis of the plant sample) and laboratory observations (viz. soil analysis and measurement of pH) are made to identify whether the symptoms appeared on plants are due to biotic or abiotic factors. This approach is an expensive, time-consuming and tedious job. To help with farming for the existing farmers and the growing generation, various systems based on Information and Communication Technology (ICT) and Artificial Intelligence (AI) is developed and incorporated into the field of agriculture as a decision support systems ([1], [2]).

TNAU AGRITECH is one of the ICT enabled agricultural initiative in Tamilnadu for agro-advisory services [3]. Still there are constraints that limit the use of ICT based systems, such as:

- illiterate farmers and lack of practical knowledge in accessing the system
- lack of effective training
- complexity in usage of ICT by the farmers
- at-times loaded with un-useful and irrelevant information
- lack of trust in ICT
- lack of broadband facility or poor network coverage, etc. [2]

From last decade, deep learning is gaining much popularity due to its performance result in accuracy, when trained with huge data. The biggest advantage of deep learning is to learn high-level features from data in a way such that it avoids the use of hand engineered features and any domain experts related to the application. Also using deep learning techniques, it takes a lot of time for training due to the large number of parameters used, but it takes less time while testing. On contrary, the traditional machine learning algorithms takes less time in training and more time in testing ([4], [5]).

The major advantages are:

- The network automatically identifies and stores the features from the training dataset.
- Robust in classification – irrespective of the scales, orientations and occlusions.

The main objectives of this paper are

- To familiarize with various existing deep learning architectures and their applications
- To compare the different architectures based on their memory size and performance
- To know how a deep learning network is created
- To project the scope of research in deep learning architectures in agriculture

Different deep learning models developed are obtained from the competitions such as Large Scale Visual Recognition Challenge (ILSVRC) and Pascal Visual Object Challenge (PASCAL VOC) instigated the researchers to develop new models or fine tune the existing models, so that the models can be applied in various applications like medical, satellite imagery, agriculture, etc. ([6], [7], [8]).

In this paper, a binary classification using a convolutional neural network and is performance analysis is presented. Later sections of the paper are organized as follows: Section II highlights the performance of the existing deep learning models in agricultural applications. Section III explains the different layers of a convolutional neural network. Section IV implements classification using a CNN from scratch. Section V concludes the aim of the study and the future scope of the proposed work

II. EXISTING DEEP LEARNING MODELS & APPLICATIONS

Deep Learning (DL) is a subset of Machine Learning (ML) which is a subset of Artificial Intelligence (AI) which plays a vital role in developing human intelligible and independent systems. DL mimics the functionality of human

978-1-7281-1380-7/19/$31.00 ©2019 IEEE

brain which consists of enormous number of neurons controlled by a central nervous system. Similarly DL also composed of several number of neural networks, where each neuron is represented as a single node and the entire activity is controlled by Central Processing Unit (CPU) or Graphics Processing Unit (GPU) ([9], [10], [11]).

Deep learning conquered many of the research areas like medical diagnosis, bioinformatics, satellite imagery applications, etc. Recently, deep learning is initiated in various applications of agriculture like disease detection and classification, plant identification and classification, fruit counting using color (RGB) images captured using mobile camera sensor either using mobile or any camera devices or fixed on any robot ([4],[5], [7], [12]).

Key components of deep learning, build for various applications, is the multilayered hierarchical representation of data in the form of a Neural Network (NN) for processing information. Among various learning approaches, currently various supervised learning models are created and deployed for image classification and detection in agriculture as well as in many other fields like video surveillage, medical diagnosis, etc.. Most of the models developed works with the principle of Convolutional Neural Network ie., extracting appropriate features automatically ([9], [10], [11]).

CNN architectures can be of various types differed in the number of layers and its topology. They are LeNet, AlexNet, CaffeNet, Simonyan and Zisserman's VGG, Inception, GoogleNet, Inception-ResNet, MobileNet ([6], [7], [8], [9], [13]).

Various deep learning models like CNN, AlexNet GoogleNet, VGG are applied for plant disease detection and diagnosis. LeNet, AlexNet, GoogleNet architecture is used by Amara et al. ans Mohanty et al for leaf disease classification utilizing the Plantvillage dataset. Pawara et al. compared the GoogleNet and AlexNet for classification purpose using the AgrilPlant, LeafSnap, Follio datsets. Among these models VGG achieved best result while classifying 17,548 test image dataset to their respective classes with an accuracy of 99.53% and very less error rate. But due to the large size of the model it is less feasible to be used in mobile or any other embedded applications ([1], [7], [8], [10]).

Like machine learning, deep learning comprise of supervised, unsupervised, semi supervised and reinforment learning methods or models. Among these models, supervised models are applied in agriculture to perform various tasks like image segmentation, image classification and object detection. The main advantage of deep learning is extracting features automatically from the raw data. Architecture of each models vary from one another based on the number of layers and various functions. The base part of every deep learning model for computer vision and image processing application is the Convolutional Neural Network (CNN). CNN have become an interesting area for researchers in computer vision ever since AlexNet popularized Deep Convolutional Neural Network (DCNN). CNN performs various phases of execution performed by different layers.

III. LAYERS OF CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Network (CNN) is a supervised deep learning technique, which has paved a revolutionary force on various computer vision and image based applications. The areas where CNN are widely used are face recognition, object detection, image classification etc. Components of a CNN model include convolutional layers, pooling layers, fully connected layers, activation functions, etc. The usages of these components vary depending on the network architecture. Fig. 1. shows the different phases of a model for predicting disease in agricultural plants using a Convolutional neural network (CNN). This model is being implemented in the proposed work.

A. Convolution Layer

The layer receives as input a RGB image or an output of another layer for further processing. Input received is read as pixel values to produce feature map representing low level features such as edges and curves. Higher level unique features can be identified through a series of more layers of convolution. Image is treated as a volume of size $W \times H \times D$. Fig. 2. shows how a color image can be represented in tensorflow and the matrix representation of each channels namely Blue, Green and Red. Training can be done by varying the 4 hyper parameters namely.

Number of filters (k): This a variable parameter. More number of filters results in more powerful model, but results in increased parameter. Convolutional filters of depth 3 are moved across the entire image to obtain an activation feature map.

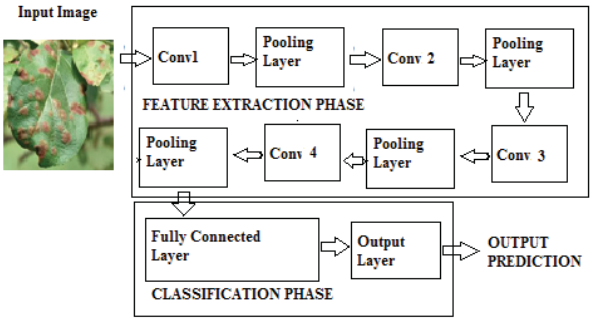

Fig. 1. Phases for Disease Prediction using CNN

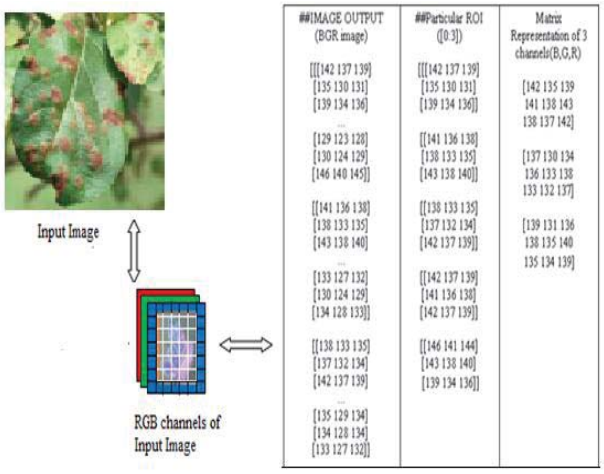

Fig. 2: Image representation in tensorflow

1) Spatial extent (filter width and height) : This varies depending on the application. Mostly used filters are 3x3, 5x5 or 7x7. For any filter of size 3x3x3 over an image of

size 32x32x3 will result in an output image of size 30x30x1. This is because 1 pixel is stripped away from left, right, top and bottom of the image.

Stride (S) : By default the kernel or filter window is slided by 1 pixel, if S=1. Similarly, the window can be slided by more than one pixel. The number of pixels slided is called stride. Bigger strides result in less overlap between the receptive fields and smaller feature map as more locations are skipped. Hence size of the feature map is shrinken than the input.

Padding: It is used to preserve the same dimensionality of feature map as that of the input by inserting some number of zero's forming the border of the image. Therefore it is also called as zero padding. The number of zero's added depends on the size of the filter chosen for convolution. If the size of the filter chosen is 3 then the number of zero's added will be 1 using the formula

$$Number\ of\ zero's = \left[\frac{K-1}{2}\right] \quad (1)$$

where K is the size of filter.

Using these hyper parameters, the receptive field size (Output Width, Output Height) and the output matrix size for both convolution and pooling layers are calculated for mapping features. Receptive field can be calculated as,

$$Output\ Width = \left(\frac{W-F_w+2P}{S_w}\right)+1 \quad (2)$$

$$Output\ Height = \left(\frac{H-F_h+2P}{S_h}\right)+1 \quad (3)$$

where, P is padding of zeros or ones, F_w, F_h, S_w, S_h represents filter width, filter height, stride width and stride height respectively.

To calculate the pooling layer output matrix,

$$Output\ Matrix, OM = \left(\frac{IM+2P-F}{S}\right)+1 \quad (4)$$

Where, OM is the output Matrix, IM is the Input matrix, P represents padding of zeros or ones, F is the filter size and S is the stride width or height. Hence the convolutional layer followed by the subsampling layers reduces the size of convolution maps and is invariant to rotation and translation.

An RGB image can be represented in matrix format or as a tensor depending on the framework used for constructing the model. A matrix is a two dimensional array of NxM numbers surrounded by brackets, where subtraction, addition are possible if the matrices are of same size and multiplying a matrix using a constant value. Multiplication of one matrix with another matrix is possible if the number of columns of first matrix is equal to the number of rows of the second matrix resulting into an another matrix. This can be represented as

$$\big((NxM)(MxP) = NxP\big) \quad (5)$$

Where, NxM and MxP are the dimensions of first and second matrix respectively, resulting in to a NxP matrix. On the other hand tensor is a generalized matrix, ie. It could be a 1-D matrix representing a vector, a 3-D matrix representing a cube of values, or a 0-D matrix representing a single number. The dimension of the tensor is called its rank.

Fig. 3. shows how a kernel or filter is applied to all channels of the input image. It combines the values of the input channels to provide an output with only 1 channel per pixel. Fig. 4. shows how a convolution is performed on a color image with a kernel of size 3x3 to obtain n feature vectors. Convolution operation is performed by sliding the filter over the input image. The area where the kernel performs operation on the image each time is called the receptive field. The size of receptive field will be same as the size of the kernel or filter. The kernel is moved over the image such that at each step it performs the weighted sum of the input pixels over which the kernel is placed.

B. *Activation Layer*

Non-linearity makes a neural network more powerful. A non-linear activation layer is applied immediately after each convolution layer. Various non linear functions are used to introduce non linearity. They are:

Tanh: This non-linearity takes the real valued number to the range [-1,1], which is mathematically represented as

$$\tanh(x) = 2\sigma(2x) - 1 \quad (6)$$

Sigmoid: This non-linearity takes the real valued number to the range between [0,1], ie. Mathematically it can be represented in the form

$$\sigma(x) = 1/(1+e^{-x}) \quad (7)$$

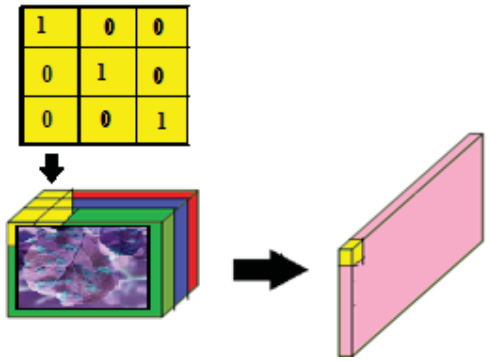

Fig. 3: Standard Convolution Process

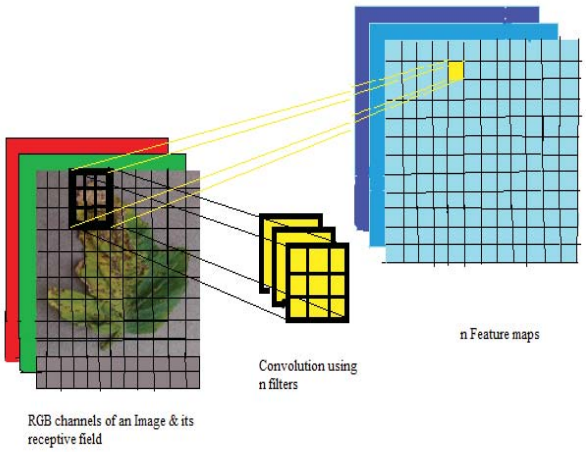

Fig. 4: Image After convolution

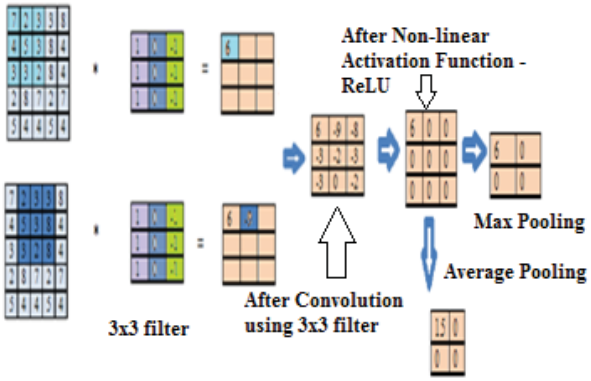

Fig. 5. Convolution- ReLU – Pooling operation example

3) Rectified Linear Unit: This increases the non linear properties of the model without changing the receptive field of the convolution layer by changing all the negative values to 0. This can be represented as

$$f(x) = \max(0, x) \tag{8}$$

C. Pooling Layer

A down sampling layer applied after activation layer to reduce the spatial dimension with no change in depth. Usually, a filter of size 2x2 is applied to the input to produce an output based on the type of pooling. Pooling can be either max pooling or average pooling where either the maximum value or the average value from every sub region bounded by the filter is taken. Pooling has no parameters.

Hence pooling layer reduces the size of the feature map [12], i.e., length and width is reduced but not the depth. This reduces the number of parameters and weights, lessens the training time and also reduces the cost of computation. This also controls overfitting.

Overfitting is a situation where the model achieves 100% or 99% on the training set but an average of 50% on the test data. This can be overcome by introducing dropout layers where a random set of activations are dropped out by setting the value to 0. Dropout is a function that improves generalization by learning several different representations of patterns. Fig. 5. shows the convolution operation where summation of the values obtained from the element wise matrix multiplication of the 2 matrices – receptive field and the kernel is performed to obtain elements in a feature map. Later ReLU and max pooling is applied to get a reduced dimension feature map [14].

D. Fully Connected Layer

This layer identifies very high level features that highly correlate to an object or class. Input to a fully connected layer is a set of features for classifying images without considering the spatial structure of the images. Most of the models use 2 fully connected layers. Output of fully connected layer is a 1D vector obtained by flattening the output of the final pooling layer. Flattening is a process of arranging 3D volume into a 1D vector. For example if 2 feature maps are generated using 2 filters are $\begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$ & $\begin{bmatrix} 1 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$ then after flattening the feature maps are arranged in a vector as

$$[1 \quad 0.5 \quad 0.5 \quad 1 \quad 1 \quad 0.5 \quad 0.5 \quad 0.5]$$

E. Classifer

Classification can be either binary class classifier or multi class classifier. For binary class classification, the activation function sigmoid is used where as for multi-class classification softmax function is used as a classifier

Mathematically softmax function can be represented as the function maps

$$\mathrm{S(a)} : R^N \rightarrow R^N \tag{9}$$

where a single element of the vector is found out using the formula,

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \forall j \in 1 \ldots N \tag{10}$$

S_j is always positive taking the value in the range (0,1), as the numerator appears in the denominator with other summation values. Its properties makes it optimal for probabilistic interpretation of an input belonging to a particular class. Hence, softmax can be interpreted as

$$S_j = P(y = j|a) \tag{11}$$

for any multiclass classification problem where y is the output class numbered 1 ... N for any N-dimensional vector a

IV. Classification Using CNN

The workflow diagram showing the experimental design of a CNN to predict whether the plant is infected or not by monitoring the leaf images is shown in Fig. 6. Different stages in processing include the following

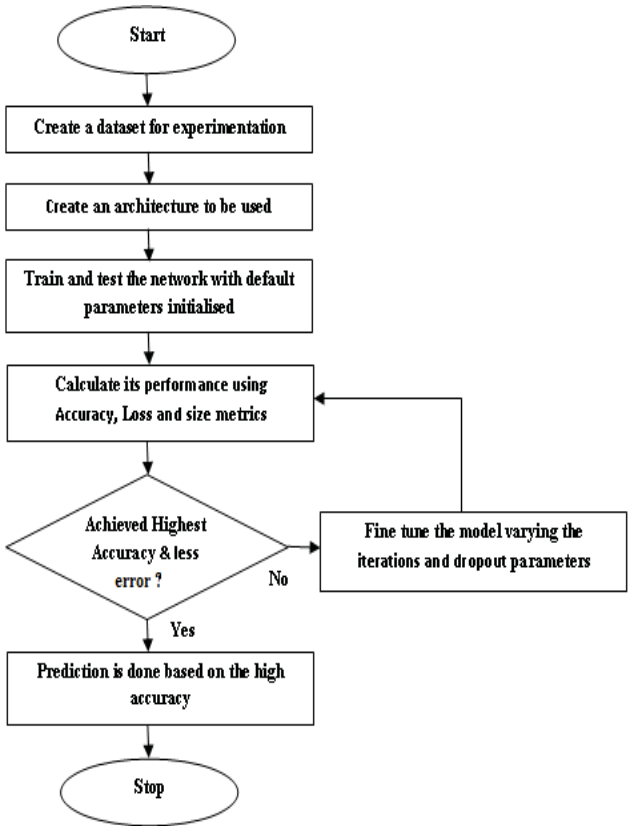

Fig. 6. Workflow Diagram for Prediction

TABLE 1. Input Image Resolution and Size Of Each Model

Architecture	Application	Input Resolution	Size of the model
LeNet-5 [4]	Classifies digits – developed for banks	32x32	70K
AlexNet [9]	Image classification	227x227	60M
ZFNet [9]	Image classification	227x227	70M
VGG [4]	Image classification	224x224	138M
GoogleNet [3]	Image classification	224x224	5M
MobileNet[11]	Image classification	224x224	4M
Proposed CNN	Image Classification	64x64	45K

A. Image dataset acquisition

Dataset can be captured manually using various digital camera, enhance and segment it if required and and save it in separate folders distinguishing the different diseased images and healthy images for different plant leaves. Images stored can be either color, grayscale or segmented images. Dataset can also be downloaded from open source websites. Usually datasets used in deep learning architectures for pre-training are ImageNet for object classification; PlantVillage for image classification and Malaya kew dataset for plant identification ([1], [7], [8], [10], [12]).

B. Preprocessing of Images

Preprocessing includes reduction in size and cropping the image to a specified size and region of interest respectively. It also enhances the image to the required color scale and is processed. Table 1. shows the input resolution of each architectures and the size of the model by calculating the number of parameters in each layer based on summation of the product of weights and biases in each layer. Similarly in the proposed CNN architecture the size of the input image is 64x64 and the size of the model is 45K. A segmented and gray scale version of the image doesn't give high performance when compared to the RGB images processing [7], [10]**.** Therefore the proposed work takes color images and is resized to 64x64 resolution for further processing.

C. Feature Extraction

The convolutional layers extract features from the resized images. When compared to general crafted feature extractors like SIFT, Gabor filter, etc., CNN learns the weights and biases of different feature maps to a specific feature extractors. The Rectified non-linear activation function (ReLu) is applied after convolution. This introduces the non-linearity to CNN ([7], [13]).

The pooling layer reduces the dimensionality of the extracted features. Different types of pooling are max pooling [4] and average pooling. Hence convolution and pooling together act as a filter to generate features. To extract features mainly it require convolution and pooling.

Convolution step operations is performed as
classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'relu'))
which indicates how a convolution layer is added using "Conv2D" function taking 4 arguments – number of filters (32), shape of each filter (3x3), shape and type of the image (eg. (64, 64, 3) indicates an RGB input image of 64x64 resolution, where 3 stands for RGB or color image) and finally an activation function used (relu stands for RectifiedLinear Unit). Pooling is added using the statement *classifier.add(MaxPooling2D(pool_size = (2, 2)))* in which a 2x2 matrix is chosen to perform operation.

D. Classification

Fully connected layers act as classifiers, where each neuron provides a full connection to all the learned feature maps obtained from the previous layer. This can be implemented using a flattening which converts all the pooled images into a continuous single dimensional vector using the following statement, *classifier.add(Flatten()).*

To implement the fully connected layer before the output layer of classification the following statements are executed *classifier.add(Dense(units = 128, activation = 'relu')),* where Dense is the function to add fully connected layer, 'units' define the number of nodes present in the hidden layer, often uses a power of 2. The final output layer can be designed based on the output generated. For binary classification output layer consist of only one node. For multiclass classification number of output nodes is based on the number of classes. Therefore for binary classification statement can be written as
classifier.add(Dense(units = 1, activation = 'sigmoid'))
where the final layer consist of only one node and uses a sigmoid activation function. Activation function used is the rectified linear unit function –ReLu. Softmax, sigmoid or Tanh functions can be used to compute the class scores. These layers utilizes the unique features learned for classification to predefine classes, or to make any numerical predictions ([7], [13]).

E. Train and Compile the Model

In every cases the entire database is randomly divided into training set and testing set following a ratio. For instance, an 80:20 ratio means 80% of the dataset belongs to training set and 20% belongs to testing set. Other splitting ratios like 70:30, 60: 40 etc. are tested to compare the performance of the model. Similarly a 20: 80 ratio also is being checked ([4],[12]).

To compile the model build, use the statement
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
where stochastic gradient descent algorithm is chosen as optimizer parameter, binary_crossentropy loss function is chosen as loss parameter and finally accuracy is the performance metric chosen as metrics parameter.

F. Testing

The compiled model is tested on a dataset which doesn't belong to the training image dataset to evaluate the accuracy of the model.

G. Implementation Results

We performed classification process with a CNN based algorithm using images of apple and tomato leafs - diseased and healthy taken from PlantVillage dataset. All input images are resized to 64x64. This resized image is fed to the convolutional layer to get the output as 62x62, using the equations (2) and (3) and to find the output width and output height (64-3+0)/1+1=62, where 64 is the input image size, 3

is the filter width (ie. 3x3) is used , 0 is the number of padding with a stride of 1.

Model is created using 4 convolution layers, where each convolution layer is followed by ReLU activation function and pooling layer. Output of the first pooling layer is 31x31 obtained as ((62-2+0)/2)+1 using equation (4) where a 2x2 window with stride 2 is used.

2 fully connected layers with softmax function is used to predict the class to which a particular image belongs to. 8000 iterations are performed on the training and testing data to get an accuracy of 74% with the loss percentage 23%. Later with repeated iterations and increasing the datset the model accuracy is improved is improved to 87%

The model overfits as there is a gap between the training and validation curves. Dropout value is set to 0.25 to reduce overfitting problem.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 31, 31, 32)	0
conv2d_2 (Conv2D)	(None, 29, 29, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 14, 14, 32)	0
conv2d_3 (Conv2D)	(None, 12, 12, 32)	9248
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 32)	0
conv2d_4 (Conv2D)	(None, 4, 4, 32)	9248
max_pooling2d_4 (MaxPooling2	(None, 2, 2, 32)	0
flatten_1 (Flatten)	(None, 128)	0
dense_1 (Dense)	(None, 128)	16512
dense_2 (Dense)	(None, 1)	129

Total params: 45,281
Trainable params: 45,281
Non-trainable params: 0

apple_healthy

Figure 7. Number of parameters obtained for CNN implemented

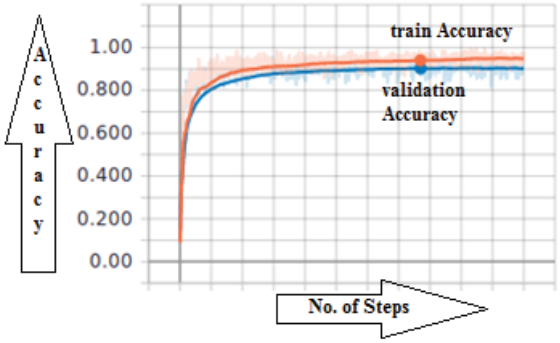

Figure 8. Accuracy after implementing Dropout

Later after using the dropout value to 0.2 and increasing the dataset the accuracy is improved to 88.7%. Fig. 7. shows the summary of the model implemented the number of parameters used which predict the size of the model. Fig. 8. shows the accuracy plotted against the number of iterations where orange color curve shows the training accuracy and blue color curve shows the validation accuracy. The model is also executed on GPU Tesla to evaluate the performance speed.

V. Conclusion And Future Work

Implemented a convolutional neural network to detect and classify whether the leaf is diseased or healthy. The achieved accuracy is 88.7 with minimum number of parameters ie. 45K when compared to other existing models. Creating and training a CNN model from scratch is a tedious process when compared to the usage of existing deep learning models for various applications to achieve maximum accuracy. So depending on the application various models can be used or retrained. This process is called transfer learning. Therefore in the future work, it is planned to utilize a model efficient than VGG and other existing architectures, such that it gives higher accuracy with minimum size and complexity, so that it can be used in mobile or any other embedded applications.

References

[1] R. Kaundal, A. S. Kapoor, and G. P. S. Raghava, "Machine learning techniques in disease forecasting: a case study on rice blast prediction.," *BMC Bioinformatics*, vol. 7, p. 485, 2006.

[2] C. Deisy and M. Francis, "Image Segmentation for Feature Extraction," no. June 2017, pp. 232–257.

[3] Saravanan R, ICT's for agricultural extension in India : Policy implications for developing countries, New India Publication Agency.

[4] A. Fuentes, S. Yoon, S. C. Kim, and D. S. Park, "A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition," *Sensors (Switzerland)*, vol. 17, no. 9, 2017.

[5] M. Brahimi, K. Boukhalfa, and A. Moussaoui, "Deep Learning for Tomato Diseases: Classification and Symptoms Visualization," *Appl. Artif. Intell.*, vol. 31, no. 4, pp. 299–315, 2017

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, C. Hill, and A. Arbor, "Going Deeper with Convolutions," pp. 1–9, 2014.

[7] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using Deep Learning for Image-Based Plant Disease Detection," *Front. Plant Sci.*, vol. 7, no. September, pp. 1–10, 2016

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst. (NIPS 2012)*, pp. 1097–1105, 2012.

[9] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.

[10] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Comput. Electron. Agric.*, vol. 145, no. September 2017, pp. 311–318, 2018.

[11] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Comput. Electron. Agric.*, vol. 147, no. July 2017, pp. 70–90, 2018.

[12] J. Amara, B. Bouaziz, and A. Algergawy, "A Deep Learning-based Approach for Banana Leaf Diseases Classification," *BTW*, pp. 79–88, 2017.

[13] Jia et al, Caffe: Convolutional architecture for feature embedding. arXiv:1408.5093, 2014

[14] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification," *Comput. Intell. Neurosci.*, vol. 2016, 2016.