IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

IDENTIFICATION AND DETECTION OF LEAFDISEASE USING DEEP LEARNING

[1]Krithiga Sukumaran krithiga.sri@gmail.com [2]Manimegalai Munisamy manimega.samy@gmail.com

[1]Assistant Professor [2]Assistant Professor,
[1]Department of ECE [2]Department of ECE
[1]ThanthaiPeriyar Government Institute of Technology,Vellore India [2]ThanthaiPeriyar Government Institute of Technology,Vellore,India

Abstract: India is an agricultural country. 70% of Indian economy depends on agriculture but leaf infection phenomena cause the loss of major crops which results in economic loss. Modern technologies have given human society the ability to produce enough food to meet the demand of more than 7 billion people. However, food security remains threatened by a number of factors including climate change, the decline in pollinators, plant diseases, and others. Plant diseases have turned into a dilemma as it can cause significant reduction in both quality and quantity of agricultural products. Automatic detection of plant diseases is an essential research topic as it may prove benefits in monitoring large field of crops, and thus automatically detect the symptoms of diseases as soon as they appear on plant leaves. The proposed system is a software solution for automatic detection and classification of tomato plant leaf diseases. To achieve this, CNN based automatic detection of disease is carried out using Village plant dataset and the accuracy if the model is found to be 98.68%.

***Index Terms*: CNN, Village plant dataset**

I.Introduction

In plants, disease is defined as any impairment of physiological function of plants, producing characteristic symptoms. Plant disease is caused by pathogens, which is defined as any agent causing disease. There are two types of agents that can destroy plants: living and non-living agents. Living agents include bacteria, fungi, insects and viruses. Non-living agents include extreme temperatures, excess moisture, insufficient nutrients, poor light, poor soil PH level and air pollutants.

Tomato is a food-rich plant, a consumable vegetable widely cultivated. The tomato, a significant contributor to reducing poverty, is seen as an income source for farm households [3]. Tomatoes are one of the most nutrient-dense crops on the planet, and their cultivation and production have a significant impact on the agricultural economy. Tomato demand is also increasing as a result of its widespread use. According to statistics, small farmers produce more than 80% of agricultural output [2]; due to diseases and pests, about 50% of their crops are lost. The diseases and parasitic insects are the key factors impacting tomato growth, making it necessary to research the field crop disease diagnosis. The manual identification of pests and pathogens is inefficient and expensive. Therefore, it is necessary to provide automated AI image-based solutions to farmers.

Due to increase in diseases among plants, cultivation of crops in order to yield quality products is extremely challenging and highly technical. This can be improved with the help of technical support. One such approach is the detection of plant diseases using deep learning. This method can efficiently detect diseases in plants with the help of a well-trained neural network. Usually, pests and diseases are found on the leaves or stems of the plant. Therefore, it is necessary for this technology to be able to identify leaves, stems, pests and diseases. It does this efficiently by using a Convolution Neural Network (CNN).

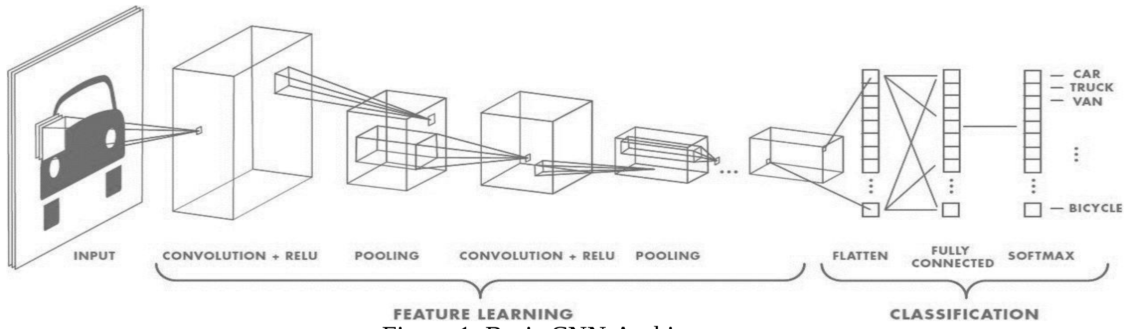

Figure 1. Basic CNN Architecture

In the recent years, deep learning in NNs has been getting much prominences. Unsupervised classification is the most active research area in hyperspectral data analysis. CNN is a leading unsupervised CNN and conventional NNs is that CNN is inspired from retinal fields in the vision system. In a simple word, CNN is an integration of biological vision and neural system. CNN is a complex architecture deep learning architecture that learns 'filters performing convolutions' in the image domain. A measure difference between which takes considerably more time to train the neurons. Nonetheless, it has remarkable classification accuracy, and rate of object recognition is very high.

In neural networks, Convolution neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used. CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output.

For image recognition problems, convolution neural networks, or CNNs, is a common choice. A common choice for the structure of a CNN consists of alternating layers of convolution and max pooling layers, which are then combined with a fully connected dense layer. The convolution layers work as a feature extractor, or a filter, and extract some sort of characteristic features from the input image. Usually, one convolution layer has several filters which are all applied in the same step to extract different features. The size of the filter, or the kernel, depends on what the size of a specific feature is expected to be extracted. Convolution Neural Network uses different layers to extract features from the input image. Those layers are,

1. Convolution layer
2. ReLU Layer (Rectified Linear Unit)
3. Pooling Layer (Max Pooling)
4. Flattening
5. Fully Connected Layer

1. Convolution layer – It is the first layer to extract features from the input image and it learns the relationship between features using kernel or filters with input images.

2. ReLU Layer – ReLU stands for the Rectified Linear Unit for a non-linear operation. The output is $f(x) = max\ (0, x)$**.** This is used because to introduce the non-linearity to CNN.

3. **Pooling Layer –** It is used to reduce the number of parameters by down sampling and retain only the valuable information to process further. There are types of Pooling: Max Pooling **A**verage and Sum Pooling

4. Flattening – This process flattens the entire matrix into a vector like a vertical one. So, that it will be passed to the input layer.

5. Fully Connected Layer – This layer passes the flatten vector into input Layer. These features are then combined to create a model. Finally, an activation function such as SoftMax or sigmoid is used to classify the outputs.

II.METHODS

In this section let see the models and datasets utilized to carry out this work. Village plant data set contains ten classes of which 9 classes are diseased and 1 class for healthy tomato leaf dataset as shown in Figure 2. In this work the entire dataset is divided into 80:20 for training and test dataset.Before training the model, image pre-processing was used to change or boost the raw images that needed to be processed by the CNN classifier. Building a successful model requires analyzing both the design of the network and the format of input data. We pre-processed our dataset so that the proposed model could take the appropriate features out of the image.

The first step was to normalize the size of the picture and resize it to 252 × 252 pixels. The images were then transformed into grey. The stage of pre-processing means that a considerable amount of training data is required for the explicit learning of the training data features. The next step was to group tomato leaf pictures by type and then marks all images with the correct acronym for the disease. In this case, the dataset showed ten classes in test collection and training.

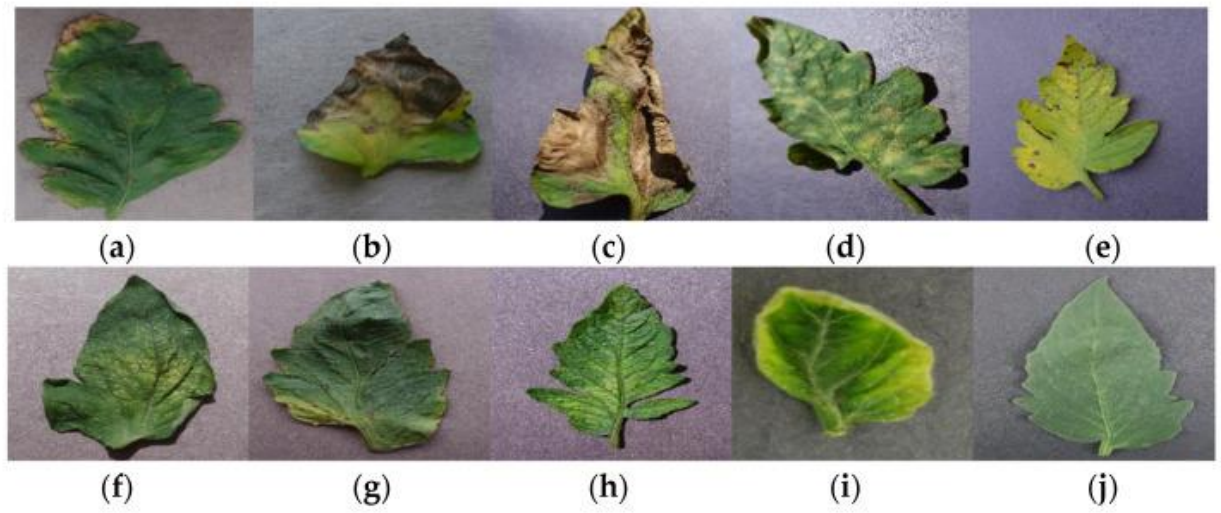

Figure 2 Sample leaf image with disease and pathogen (a) Bacterial_Spot (*Xathomonas vesicatoria*) (b) Early_Blight (*Fungus Alternaria solanji*) (c) Late_Blight (*Phytophthora infestans*) (d) Leaf_Mold (*Clasospporium fulvum*) (e)Septoria_leaf (*Fungus Septoria lycopersci*) (f) Spider_Miters (*Floridana*) (g) Target_Sport (*Fungus Corynespora*) (h) Tomato_Mosaic_Virus (*Tobarnovirus*) (i) Tomato_Yellow_Leaf_Curl_Virus (*genus Begomovirus*) (j) Healthy_Leaf

III. Convolutional Layers

3.1 Layer1

In this layer, a 2D convolutional layer is added to process the two-dimensional input images. The first argument passed to the Conv2D() layer function is the number of output channels. In this case 32output channels are used. The next input is the kernel size, which in this case is chosen a 5×5 moving window, followed by the strides in the *x* and *y* directions (1, 1). Next, the activation function, which is a rectified linear unit (ReLU) and finally the model with the size of the input to the layer. Declaring the input shape is only required for the first layer. The output of Layer 1 produces an output of size 252x252x32.

Next, a 2D max pooling layer is added. In max pooling layer no hyperparameters that is to be learnt. Only it shrinks the output .In this case, the size of the pooling in the *x* and *y* directions is 3 x 3 and the strides are not declared as the default value of strides is 1. This pooling layer produces a max-pooled output with an image size of 84 x 84.

3.2 Layer2

In next layer another convolutional and max pooling layer is added, with 64 output channels. The default strides argument in the Conv2D() function is (1, 1) in Keras, so it need not to be declared again. Usually, the default strides argument in Keras is to make it equal or the pool size. The next input is the kernel size, which in this case we have chosen a 3 x 3 moving window, followed by the strides in the x and y directions (1, 1). Next, the activation function, which is again a rectified linear unit (ReLU).

The input tensor for this layer is (84, 84, 32). The 84 x 84 is the size of the image, and the 32 is the number of output channels from the previous layer. This allows rapid assembling of network architectures irrespective of the sizes of the tensors flowing around our networks.

Next, a 2D max pooling layer is added. In this case, the size of the pooling in the *x* and *y* directions is 2 x 2. This pooling layer produces a max-pooled output with an image size of 41x41.

3.3 Layer 3

In this layer another set of convolutional and max pooling layer is added with 128 output channels with a kernel size of 3 x 3 followed by an activation function, which is ReLU. The input tensor for this layer is (41, 41, 64) in which the 41 x 41 is the size of the image, and the 64 is the number of output channels from the previous layer. Next, a 2D max pooling layer is added with pooling size of 2 x 2 which produces a max-pooled output with an image size of 19x19.

3.4 Fully Conncected Layer

After defining the convolutional layers, the fully connected layers are added. To connect the output of the pooling layer to the fully connected layer, the previous layer output needs to be flattened into a single (N x 1) tensor.

Here, the previous layer is the pooling layer that produces an output of 128 channels of 19 x 19 pooling matrices. All of this output data is flattened into a vector with one column and 19 x 19 x 128 = 46208 rows. This equates to 46208 nodes per training sample.

3.5 Dense Layer

After flattening the output, a dense layer is created with 512 nodes with ReLU as an activation function. Then the connection is made between the flattened output and 512 nodes which produces a fully connected layer. The weights of the fully connected layer are determined by multiplying them with the flattened convolutional output, then adding a bias. The total parameters in this layer is the value which is obtained when multiplying the rows of the flattened data with the total number of dense nodes. In this case, a total of 23659008 learning parameters are created.

3.6 Output Layer

This layer is also a dense layer that connects to the output, and therefore a soft-max activation is used to produce the predicted output values.

This dense is declared with the value that is equal to the number of diseased classes in the dataset. Here, totally 10 classes are in the dataset. So, a dense layer with 10 nodes is created. This layer with 10 nodes is connected to the previous dense layer with 512 nodes that approximately produces about 5120 training parameters. The total learning hyperparameters are given in the table 1

Table 1 Hyperparameters involved in training

Layer (Type)	Output shape	Parameters
conv2d_1 (Conv2D)	(None,252, 252, 32)	2432
max_pooling2d _1	(None, 84, 84, 32)	0
conv2d_2------ (Conv2D)	(None, 82, 82, 32)	18496
max_pooling2d _2	(None, 41, 41, 64)	0
conv2d_3 ------- ----(Conv2D)	(None, 39, 39, 128)	73856
max_pooling2d _3	(None, 19, 19, 128)	0
flatten_1 -------- ----(Flatten)	(None, 46208)	0
dense_1 --------- ----(Dense)	(None, 512)	23659008
dropout_1	(None, 512)	0
dense_1 --------- ----(Dense)	(None, 10)	5130
Total parameters		23758922

3.7 Training the Network

After building the CNN architecture, it needs to be trained with the dataset in a particular learning method that suits the problem. Since the dataset is already classified under different classes, supervised learning method is followed while training the CNN architecture. To improve the accuracy and efficiency of the network in predicting the leaf disease the weights of the network needs to be updated after every epoch in accordance with the output. So, the back propagation method is followed to do the same. Also, the other essential parameters for the network training is chosen.

- Loss – The loss function represents the inaccuracy in predictions made by the neural network
- Optimizer **–** It update the weight parameters to minimize the loss function.
- Metrics – Accuracy is a common metric for binary classifiers. It takes into account both true positives and true negatives with equal weight.
- Epoch – It is the number of times that the CNN should be trained with the training dataset.
- Learning rate – It decides the rate of learning.

In order to train the CNN, the above hyperparameters must be defined. The CNN training includes the following steps.

1. Chose a loss function and an optimizer
2. Create correct prediction and accuracy evaluation operations
3. Initialize the operations
4. Determine the number of batches runs within a training epoch
5. For each epoch:
    - Extract the batch data
    - Run the optimizer and cross-entropy operations
    - Add to the average cost

6. Calculate the current test accuracy
7. Calculate the final test accuracy.

In this case, the categorical cross entropy is used as the loss function. This is chosen because the detection of leaf disease comes under categorical class classification type.
For optimizer, Adam optimizer is used with a learning rate of 0.001.

Once the parameters are chosen, the model is trained using model.fit() function During the training process, for each step, one data object is run through the model and the weights are adjusted. When all training data has passed through the network once, one epoch is completed.

IV. TESTING AND EVALUATION

Evaluation is the process of measuring the accuracy of the trained network. The 80% of the dataset is used for the training and the remaining 20% is used for evaluation and for validating the trained network. The testing data is also preprocessed as same as training data. After the completion of the training and testing processes, the proposed deep CNN model was selected for the validation process. In addition, the trained model was tested with new inputs and the results are verified. The achieved validation loss and validation accuracy of the model is about 13.56% and 98.68% respectively.

V. RESULTS &CONCLUSIONS

The proposed Deep CNN model can effectively classify 10 distinct classes of healthy and diseased tomato plant using leaf images. The Deep CNN model was trained and tested with using an augmented dataset with 18,835 images and 20 training epochs. The proposed model achieves an average validation accuracy of 98.68% in the classification of the testing set plant leaf images. The number of training epochs, batch size and dropout had greater influences on the respective results. The most important goal of the future work will be to extend the plant disease identification objective from plant leaves to other parts of the plants, such as flowers, fruits and stems.

REFERENCES

[1] Bharate AA, Shirdhonkar MS. A review on plant disease detection using image processing. In: 2017 International Conference Intelligence Sustain. System; 2017. p. 103–9. DOI: 10.1109/ISS1.2017.8389326.
[2] Stilwell M. The global tomato online news processing in 2018.
[3] Wang R., Lammers M., Tikunov Y., Bovy A.G., Angenent G.C., de Maagd R.A. The rin, nor and Cnr spontaneous mutations inhibit tomato fruit ripening in additive and epistatic manners. *Plant Sci.* 2020; **294**:110436–110447. doi: 10.1016/j.plantsci.2020.110436.
[4] Bhumika S.Prajapati, Vipul K.Dabhi Harshadkumar, B.Prajapati, "A Survey on Detection and Classification of Cotton Leaf Diseases", International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) – 2016.
[5] Ebrahimi MA, Khoshtaghaza MH, Minaei S, Jamshidi B. Vision-based pest detection based on SVM classification method. Comput. Electron. Agric. 2017; 137:52–8. DOI:10.1016/j.compag.2017.03.016.
[6] El Houby EMF. A survey on applying machine learning techniques for management of diseases. J Appl Biomed 2018;16(3):165–74. DOI: 10.1016/j.jab.2018.01.002.
[7] Elham Omrani, Benyamin Khoshnevisan, Shahaboddin Shamshirband, Hadi Saboohi, Nor Badrul Anuar, Mohd Hairul Nizam Md Nasir "Potential of radial basis function-DOI: 10.1016/j.measurement.2014.05.033.
[8] Johannes A, Picon A, Alvarez-Gila A, Echazarra J, Rodriguez- Vaamonde S, Navajas AD, et al. Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case. Comput. Electron. Agric.2017; 138:200–9. DOI: 10.1016/j.compag.2017.04.013.
[9] Kulkarni, O. (2018). Crop Disease Detection Using Deep Learning. 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA). DOI:10.1109/iccubea.2018.8697390

Krithiga Sukumaran She received her B.E degree in Electronics and Communication Engineering from Madras University and M.E degree in Applied Electronics from College of Engineering, Guindy. Currently working as a Assistant Professor in Thanthai Periyar Government Institute of Technology,Vellore since April 2013 and pursuing the research work in the field of Compressed Sensing for Image Processing Applications in Anna University,Chennai

ManimegalaiMunisamy.She received her B.E degree in Electronics and communication Engineering from Thanthai Periyar Government Institute of technology,Vellore and M.E degree in Communication systems from College of Engineering,Guindy,India.She is currently working asAssistant Professor in Thanthai Periyar Government Institute of Technology,Vellore,India.Her field of interest includes Wireless communication and Wireless Sensor networks. She is currently pursuing her Ph.D in the field of Wireless communication.