IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 3, March 2022

DOI: 10.17148/IJARCCE.2022.11326

Identifying Plants Diseases and providing supplements - using CNN model

Shailendra Singh, Rishab Jain, Rishabh Tripathi, Riya Goel,Vanshika Rastogi

Inderprastha Engineering College, Department of Computer Science and Engineering,

AKTU university, India, Uttar Pradesh

Abstract: Plants and crops that are afflicted by pests or diseases have an impact on agricultural productivity. Generally, farmers and professionals examine plants with their naked eyes in order to discover and identify illness. However, this procedure is time-consuming and frequently wrong. Data augmentation and picture pre-processing techniques are used to detect plant diseases, resulting in faster and more reliable findings. The purpose of this study is to present a novel method to the construction of a disease recognition model using CNN, which supports plant leaf image classification utilising convolutional networks and the Deep Learning algorithm. Advances in technology allow for the expansion and enhancement of plant protection practises, as well as development in the computer vision sector and using machine learning applications in the world of agriculture and farming, making it easier and more successful with a completely unique training method. All of the necessary steps and modules for implementing the plants disease recognition model are fully described throughout the paper, beginning with image collection to create a database, which will be evaluated by agricultural experts, and a deep learning algorithm framework to perform CNN training. Using the deep convolutional neural network that we will train, test, and validate, this technique paper might be a novel strategy to detecting and identifying plant illnesses. The created CNN model's development and innovation are shown in its simplicity; healthy leaves and backdrop pictures are consistent with previous CNN models, Using CNN, the model is able to discriminate between damaged and healthy leaves. Plants are the world's primary food source. Plant infections and illnesses are a significant hazard, and the most frequent method of diagnosing plant diseases is to examine the plant body for visible signs and growth [1]. Different research efforts intend to identify realistic techniques to plant protection and support our farmers as an alternative to the old time-consuming process. In recent years, technological advancements have spawned a slew of new ways to complement old procedures [2]. In picture classification challenges, deep learning approaches are particularly powerful and successful.

Keywords: Plant's Leaf Disease, CNN model (Deep Learning Algorithm)

INTRODUCTION

People can now supply the right nourishment, nutrients, and food to suit the demands of the world's rising population thanks to advancements in technology. When it comes to India, 70 percent of the population is directly or indirectly involved in cultivating land or agricultural, which is a significant figure. Modern technology has enabled us to produce more food than ever before in order to fulfil the needs of over 7 billion people. Plant diseases pose a worldwide danger to food security, but they can have devastating repercussions for small-scale farmers whose livelihoods are dependent on healthy crops and farming. Small-scale farmers in rural regions have no other activity than farming, and if their crops are damaged, they have no other means of surviving. Small-scale farmers account for more than 80% of agricultural production in the developing countries, with pests and illnesses accounting for 50% of yield losses.

Various approaches to preventing crop loss due to diseases and pests have been tested. The widespread use of pesticides on crops has been progressively complemented by Integrated Pest Management (IPM) measures, which include properly recognizing a crop or plant disease based on its appearance
for effective disease control of plants or crops our agricultural extension agencies or other institutions involved in the area of agriculture, such as local plant clinics, used to identify crop and plant diseases in the past. In the modern day, such efforts are aided by the availability of internet information for illness detection, and global Internet adoption is expanding. Even more recently, tools based on mobile phones have seen a tremendous surge in popularity, owing to the world's unprecedented quick adoption of mobile phone technology.

Smart devices offer very effective and efficient approaches to help us identify plant and crop diseases due to their increased power, with more pixels per inch and high quality and crisp images of leaves, and extensive built-in sets of accessories, such as advanced cameras with much higher resolution. By the end of 2015, 69 percent of the worldwide

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 3, March 2022

DOI: 10.17148/IJARCCE.2022.11326

population had access to smart devices with wideband coverage, and subscriptions had more than doubled since 2007. The combination of global subscription penetration, HD sophisticated cameras, and boosted CPUs in smart devices leads to the assumption that plant and agricultural diseases are identified using automated picture identification and pre-processing or data augmentation. Here, we demonstrate technical usefulness by employing a deep learning approach – CNN model, with a dataset titled "plant village dataset," which contains images in 39 different classes of different plants and background images, with each class containing nearly 61486 images of leaves of different plants predicting whether they are healthy or not. To increase the size of our data collection, we are employing six Augmentation approaches.

Image flipping, gamma correction, noise injection, PCA color augmentation, rotation, and scaling are the six data augmentation techniques utilized in diagnosing plant diseases. In Noisy injection, we will generate random noise and a variable called "noise," and then add noise to our dataset (dataset = dataset + noise). The noisy dataset is then divided into three parts: 70 percent for training [1], 15 percent for validation [2] , and 15 percent for testing [3].

CNN Model, a deep neural network model, has been effectively implemented in every discipline, including end-to-end learning, image classification, data augmentation, and so on. Neural networks translate an input, such as a picture of a damaged plant, to an output, such as a crop-disease pair. A neural network's nodes are mathematical functions that accept numerical inputs from the incoming edges and provide a numerical output as an outgoing edge. Here, we're extracting features with CNN and utilizing them to build our model, which we'll train further. We are defining the filter size for the conv layer and the pool layer, as well as the form of each layer. Then, we employ the torch framework to train, validate, and test our model. The difficult element is creating and deploying a deep learning CNN model in such a manner that both the network structure and the modules, which are the nodes and edge weights, accurately transfer the input to the output. Deep neural CNN networks are taught by fine-tuning the network parameters so that the mapping improves with time. This is a difficult process that has been enhanced by a variety of conceptual and engineering advances.

MATERIALS AND METHODS

"Plant Village" data set:

There are 39 different kinds of plant leaf and background photos accessible in the plant village dataset. Each class has 61486 photos in the collection. To increase the size of the dataset, we are applying six distinct data augmentation approaches. Image flipping, Gamma correction, noise injection, PCA colour augmentation, rotation, and scaling are the data augmentation techniques employed.

Image Pre-processing :

Pre-processing photographs often include removing low-recurrence foundation turmoil, normalising the power of individual particle pictures, removing reflections, and veiling segments of images. Picture pre-processing is an information-improvement approach. Furthermore, the photo pre-processing approach includes physically editing the seeming variety of pictures, producing the square around the leaves, to showcase the district of curiosity (plant leaves). Pictures having a more modest purpose and measurements that were not exactly 500 pixels were not considered significant for the dataset throughout the collection period. Furthermore, only the images where the location of interest was in higher the aim were designated as a qualifying possibility for the dataset. As a result, it was ensured that images had all of the necessary data for highlight learning. Numerous assets may be discovered by searching the Internet; yet, their meaning is sometimes troublesome. Considering a real worry for confirming the exactness of classes in the dataset, which was first produced by a keyword search, horticulture professionals examined leaf images and labelled all of the images with appropriate illness abbreviations. As is well known, it is critical to use accurately defined images for the dataset development and approval. Only in this way can a suitable and strong identification model be constructed. During this step, copied images that remained after the underlying focus of grouping photographs into classes were removed from the dataset.

Training of Neural Network using CNN :-

```
indices = list(range(len(dataset)))

split = int(np.floor(0.85 * len(dataset)))   # train_size

validation = int(np.floor(0.70 * split))   # validation

np.random.shuffle(indices)

train_indices, validation_indices, test_indices = (
    indices[:validation],
    indices[validation:split],
    indices[split:],
)
```

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 3, March 2022

DOI: 10.17148/IJARCCE.2022.11326

In the preceding code, we first gather indices and then divide them into train, test, and validate. The remaining photos are for testing, with a total of 36584 for train, 15679 for validation, and 15679 for validation.

```
train_sampler = SubsetRandomSampler(train_indices)
validation_sampler = SubsetRandomSampler(validation_indices)
test_sampler = SubsetRandomSampler(test_indices)
```

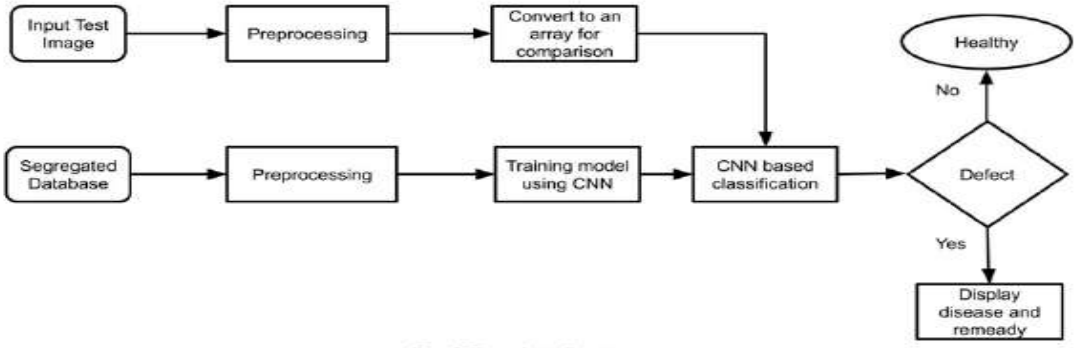
To sample our data, we utilize SubsetRandomSampler. We are constructing a SubsetRandomSampler Object here, and we will utilize this sampler in the train and test data loaders later.

```
batch_size = 64

train_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=train_sampler
)

test_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=test_sampler
)

validation_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=validation_sampler
)
```

Train sampler is used for train-loader and vice versa. Our plant village dataset is now available for training and testing...

Flow Diagram:

In the following phase, the input test picture is created and pre-processed before being turned into array form for difference. The selected database is segregated and pre-processed before being retitled into appropriate directories. The CNN model is used to train the model, and then classification takes place. The presentation of the outcome tracks the evaluation of the test image and the training model. If the plant has a fault or infection, the box depicts the ailment as well as the treatment.



Dig. 2 Flow chart diagram

Figure 1: Infected Plant Leaves

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 3, March 2022

DOI: 10.17148/IJARCCE.2022.11326

LITERATURE SURVEY

S. Khirade et al. [1] used digital image processing techniques and a back propagation neural network (BPNN) to solve the challenge of plant disease detection in 2015. The authors developed various strategies for detecting plant illness using photographs of leaves. They used Otsu's thresholding technique, followed by border detection and spot detection, to segment the contaminated region of the leaf. Following that, they retrieved parameters such as colour, texture, morphology, edges, and so on for plant disease categorization. BPNN is used for classification, that is, to identify plant disease.

In this study, ShiroopMadiwalar and Medha Wyawahare examined several image processing algorithms for plant disease identification [2]. The authors looked at colour and textural traits to detect plant illness. They tested their algorithms on a collection of 110 RGB photos. The mean and standard deviation of the RGB and YCbCr channels, the grey level cooccurrence matrix (GLCM) features, and the mean and standard deviation of the picture convolved with the Gabor filter were retrieved for classification. For classification, a support vector machine classifier was utilised. The authors came to the conclusion that GCLM characteristics are good in detecting normal leaves. Color features and Gabor filter features, on the other hand, are thought to be the best for identifying anthracnose infected leaves and leaf spot, respectively. Using all of the retrieved data, they reached the greatest accuracy of 83.34 percent.

Peyman Moghadam et al. demonstrated the use of hyperspectral imaging in the identification of plant diseases [3]. This study made use of the visible and near-infrared (VNIR) and short-wave infrared (SWIR) spectrums. For leaf segmentation, the authors employed the k-means clustering approach in the spectral domain. To remove the grid from hyperspectral pictures, they suggested a unique grid removal technique. The authors attained an accuracy of 83 percent using vegetation indices in the VNIR spectral band and 93 percent using the whole spectrum. Though the suggested technique achieved greater accuracy, it necessitates the use of a hyperspectral camera having 324 spectral bands, making the solution prohibitively expensive.

Sharath D. M. et al. created a Bacterial Blight detection method for the Pomegranate plant by using characteristics such as colour, mean, homogeneity, SD, variance, correlation, entropy, edges, and so on. The authors used grab cut segmentation to segment the region of interest in the picture [4]. The edges in the photos were extracted using the Canny edge detector. The authors have successfully built a method that can predict the degree of infection in the fruit.

To identify plant illness, Garima Shrestha et al. used a convolutional neural network [5]. The authors categorised 12 plant diseases with 88.80 percent accuracy. Experimentation was carried out using a dataset of 3000 high quality RGB photographs. The network is made up of three blocks of convolution and pooling layers. As a result, the network is computationally costly. In addition, the model's F1 score is 0.12, which is relatively low due to a larger number of incorrect negative predictions.

"Leaf Disease Detection and Grading Using Computer Vision Technology and Fuzzy Logic," Akanksha Rastogi, Ritika Arora, and Shanu Sharma. K-means clustering is utilized to split the defective region; GLCM is used to extract textural information; and Fuzzy logic is used to grade diseases. They employed an artificial neural network (ANN) as a classifier to determine the degree of the damaged leaf.

Uan Tian, Chunjiang Zhao, Shenglian Lu, and Xinyu Guo, "SVM-based Multiple Classifier System for Recognition of Wheat Leaf Diseases," Color characteristics are transferred in RGB to HIS, and seven invariant moments are used as form parameters by utilising GLCM. They utilised an SVM classifier with MCS to identify illness in a wheat plant offline.

P. R. Rothe and R. V. Kshirsagar proposed a "Cotton Leaf Disease Identification Using Pattern Recognition Techniques" that employs snake segmentation, with Hu's moments serving as a distinguishing feature. The active contour model is utilised to confine the vitality inside the infection area, and the BPNN classifier deals with the various class difficulties. The average categorization rate is 85.52 percent.

PROPOSED METHOD

Plants are prone to a variety of disease-related illnesses and convulsions. There are several reasons that may be distinguished by their impact on plants, disruptions caused by environmental factors such as temperature, humidity, excess or inadequate food, light, and the most prevalent illnesses such as bacterial, viral, and fungal infections. We employ the CNN algorithm in the suggested system to identify illness in plant leaves since it achieves the highest accuracy if the data is good.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 3, March 2022

DOI: 10.17148/IJARCCE.2022.11326

A. Dataset

This collection contains 39 distinct kinds of plant leaf and background photos. There are 61,486 photos in the collection. To increase the size of the data-set, we applied six distinct augmentation approaches. Image flipping, Gamma correction, noise injection, PCA colour augmentation, rotation, and scaling are among the techniques used. There are a total of 39 Classes to forecast using the CNN Model.

Plant	Disease Name	No. of Images
Apple	Healthy	2008
	Diseased Scab	2016
	Diseased: Black rot	1987
	Diseased: Cedar apple rust	1760
Corn	Healthy	1859
	Diseased: Cercospora leaf spot	1642
	Diseased: Common rust	1907
	Diseased: Northern Leaf Blight	1908
Grapes	Healthy	1692
	Diseased: Black rot	1888
	Diseased: Esca (Black Measles)	1920
	Diseased: Leaf blight (Isariopsis)	1722
Potato	Healthy	1824
	Diseased: Early blight	1939
	Diseased: Late blight	1939
Tomato	Healthy	1926
	Diseased: Bacterial spot	1702
	Diseased: Early blight	1920
	Diseased: Late blight	1851
	Diseased: Leaf Mold	1882
	Diseased: Septoria leaf spot	1745
	Diseased: Two-spotted spider mite	1741
	Diseased: Target Spot	1827
	Diseased: Yellow Leaf Curl Virus	1961
	Diseased: Tomato mosaic virus	1790

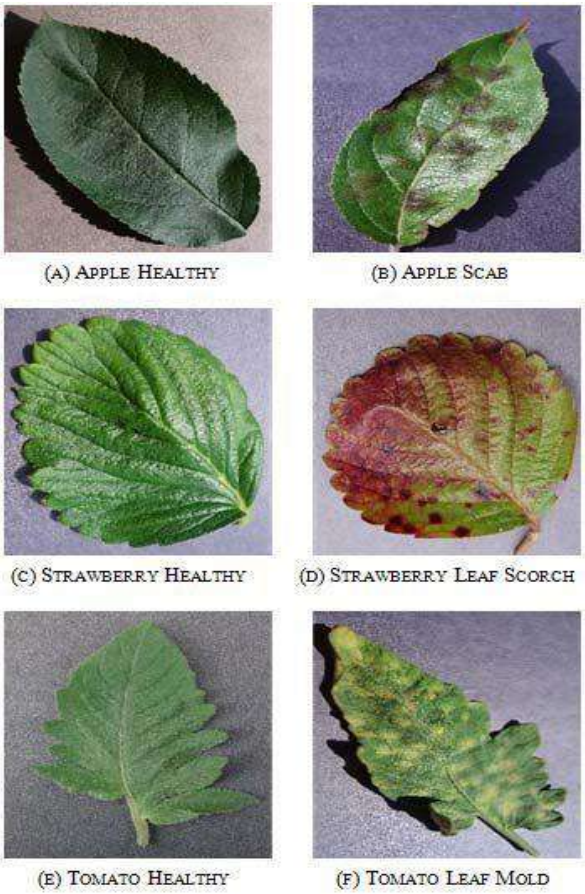
Fig:2 DATASET BREAKUP (TABLE I)
DATASET IMAGES

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 3, March 2022

DOI: 10.17148/IJARCCE.2022.11326



(A) Apple Healthy (B) Apple Scab

(C) Strawberry Healthy (D) Strawberry Leaf Scorch

(E) Tomato Healthy (F) Tomato Leaf Mold

B. Performance Evaluation

We run all of our experiments across a range of train-test set splits to get a sense of how our approaches will perform on new unseen data, as well as to keep track of if any of our approaches are overfitting. These splits are 80–20 (80% of the whole dataset used for training, and 20% for testing), 60–40 (60 percent of the whole dataset used for training, and 40% for testing), and 50–50 (50 percent of the whole dataset used for training, and 50% for testing (20 percent of the whole dataset used for training, and 80 percent for testing).

shape = (channels, height , width)

In PyTorch, shape is not automatically calculated we manually have to take care of shape on each layer. At the First Fully connected layer we have to mention output size as per the shape of the convolutional layer. This calculation is also called Convolutional Arithmetic.

Here is Equation for Convolutional Arithmetic:

Shape:

- Input: $(N, C_{in}, H_{in}, W_{in})$
- Output: $(N, C_{out}, H_{out}, W_{out})$ where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

Dilation = 0, for this project.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 3, March 2022

DOI: 10.17148/IJARCCE.2022.11326

B.Data Augmentation

•Image Flipping - Depending on the item in the image, the image can be flipped horizontally or vertically. It's a more straightforward method of increasing the performance of our trained DL model. It also allows us to flip images in the left-right and up-down directions.
•Gamma Correction - This adjusts the image's overall brightness. Changing the amount of Gamma Correction affects not just the brightness, but also the red-green-blue ratios (RGB).
•Noise Injection - Adding noise to a model during training can increase the network's resilience, resulting in improved generalisation and faster learning. (Using Numpy in Python, add noise to a single).
- Create random noise using a variable called 'noise.'
- Make Dataset noisier (Dataset = Dataset + noise).
- Divide the Noise Dataset into three sections: -
1) 70% of the time is spent on training.
2) Validation is worth 15%.
3) 15% is set aside for testing.
• PCA Color Augmentation - Atter RGB channel intensities following natural picture variances.
• Rotation - A source picture is rotated by a random number of degrees clockwise or anticlockwise, altering the location of an item in the frame, ranging from 0 to 360 degrees.
• Scaling - We choose a small picture size within a certain dimension range at random.

We're utilising Pytorch in our project, and we're using transformations for Data Augmentation, which acts as a filter for all photos.

C.Image Acquisition

This is the method of obtaining photos using a camera by visiting the location or using other accessible resources such as image databases or internet repositories. The pictures are taken in three colours: red, green, and blue (RGB), for which a colour transformation structure is developed and a device independent colour space transformation is implemented.

Data Augmentation

- **Image Flipping:-** Depending on the object in the image, the image can be flipped horizontally or vertically. It's a more straightforward method of increasing the performance of our trained DL model. It also allows us to flip images in the left-right and up-down directions.
- **Gamma Correction:-**Gamma correction technique regulates the brightness of image.Changing the amount of gamma correction affects not only the brightness, but also the red-green-blue ratios (RGB).
- **Noise Injection:-**By adding noise during the training can improve robustness of network.This will result in better generalization and faster learning.Noise injection can be done by:-
- Create random noise using the variable 'noise.'
- Increase the noise in Dataset (Dataset = Dataset + noise).
- Divide the Noise Dataset into three parts: -
- 70% for instruction.
- Validation is worth 15% of the total.
- 15% is allotted for testing.

- **PCA Color Augmentation:-** Change the intensity of the RGB channels as the image changes naturally**.**
- **Rotation:-** A random rotation of a source image clockwise or anticlockwise by some number of degrees, changing the position of an object in the frame, ranging from 0 to 360 degrees, is performed.
- **Scaling:-**Then we randomly pick a short size image with dimension range.

In our project, we are using pytorch in which transforms are used which works as a filter for all images.

Image Acquisition

Image acquisition is the process in which image is taken as input given by clicking a picture of an image or we can also give image input from any other available sources. The images are in three colors red, green and blue(RGB) for which a transformation structure is created.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

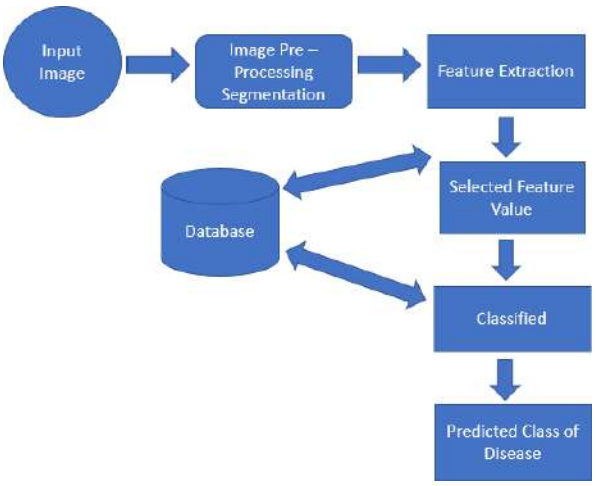
International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 3, March 2022

DOI: 10.17148/IJARCCE.2022.11326

Image Pre-processing

Firstly we should have indices and then divide the data into train, test and validation data. There will be 36584 images for train,15679 images for validation and remaining images for training. Then we create an object of SubsetRandomsampler which is generally used to sample our data and later we use this sampler to train and test the data loader.



SYSTEM OVERVIEW

Feature Selection

Feature selection is a critical step in all machine learning problems. For model creation, we employ a convolutional neural network. In this size of fiter of conv layer and pool layer and shape of each layer is also specified. In the model, we use ReLU as the activation function to remove nonlinearity, followed by Batch Normalization to normalise the neuron weights, but for the final layer, we must use SoftMax activation. In PyTorch, we have a cross-entropy loss that is a hybrid of SoftMax and category cross-entropy loss. Batch Gradient Descent - batch gd() is the function where all of the learning took place.

Classification Algorithm

For classification, CNN (convolutional neural network) was employed. The training dataset accounts for 80% of the total, while the testing dataset accounts for the remaining 20%. There are 56,236 images in the training dataset and 14,059 images in the testing dataset. The model was trained using 56,236 images, of which 14,056 were left unseen by the model in order to test its accuracy. Following fine-tuning, a pre-trained VGG16 model achieved roughly 87 percent on Train data, 84 percent on Validation data, and 83 percent on Test data. VGG-16 is composed of three main components: convolution, pooling, and fully connected layers.

- Convolution layer- Filters are applied to images in this layer to extract features. The size of the kernel and stride are the most important parameters.
- Pooling layer- Its purpose is to reduce the spatial size of a network in order to reduce the number of parameters and computations.
- Fully Connected- These are connections to previous layers that are fully connected, as in a simple neural network.

Flask Web App

After we've created this model, we'll build a flask web application and deploy it to the cloud.

EXPERIMENTATION AND RESULT

We used 39 different plant leaf and background image classes. There are 61,486 images in the data set. To increase the size of the data-set, we used six different augmentation techniques. Image flipping, Gamma correction, noise injection, PCA colour augmentation, rotation, and scaling are among the techniques used. We also use Transforms for Data Augmentation, such as cropping the image, resizing the image, converting the image to tensor, rotating the image, and many more.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 3, March 2022

DOI: 10.17148/IJARCCE.2022.11326

Essentially, we start by resizing each image to 224 x 224. This image is then fed into the Convolutional Neural Network. We feed a colour image, so it has three RGB channels. In the first conv layer, we use 32 filter sizes or output channels.That is, we apply 32 different filters to the images in order to find features, and then we create a features map with 32 channels using those features. As a result, 3 x 224 x 224 becomes 32 x 222 x 222. Following that, we use the ReLU activation function to remove nonlinearity, followed by Batch Normalization to normalise the neuron's weights. Following that, we feed this image to the max pool layer, which only takes the most relevant features, resulting in an output image with the dimensions 32 x 112 x 112. The image is then fed to the next convolutional layer, which follows the same procedure as described above. Finally, we flatten the final max pool layer output and feed it to the next linear layer, also known as a fully connected layer, before predicting 39 categories as a final layer. As a result, the model output is tensor 1x39 in size. And we take an index of the tensor's maximum value from that tensor. Our main prediction is based on that particular index.

By Using this Current Model we are getting accuracy 87 % on Train data, 84 % On Validation data and 83% on Test data.

Dataset Link (Plant Vliiage Dataset):
https://data.mendeley.com/datasets/tywbtsjrjv/1 (https://data.mendeley.com/datasets/tywbtsjrjv/1)

In [4]:

```
transform = transforms.Compose(
    [transforms.Resize(255), transforms.CenterCrop(224), transforms.ToTensor()]
)
```

```
<IPython.core.display.Javascript object>
```

In [5]:

```
dataset = datasets.ImageFolder("Dataset", transform=transform)
```

```
<IPython.core.display.Javascript object>
```

```
dataset
```

Out[6]:

```
Dataset ImageFolder
    Number of datapoints: 61486
    Root location: Dataset
    StandardTransform
Transform: Compose(
               Resize(size=255, interpolation=bilinear, max_size=None, antia
lias=None)
               CenterCrop(size=(224, 224))
               ToTensor()
           )
```

```
<IPython.core.display.Javascript object>
```

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 3, March 2022

DOI: 10.17148/IJARCCE.2022.11326

```
class CNN(nn.Module):
    def __init__(self, K):
        super(CNN, self).__init__()
        self.conv_layers = nn.Sequential(
            # conv1
            nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.Conv2d(in_channels=32, out_channels=32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.MaxPool2d(2),
            # conv2
            nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(64),
            nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(64),
            nn.MaxPool2d(2),
            # conv3
            nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(128),
            nn.Conv2d(in_channels=128, out_channels=128, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(128),
            nn.MaxPool2d(2),
            # conv4
            nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(256),
            nn.Conv2d(in_channels=256, out_channels=256, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(256),
            nn.MaxPool2d(2),
        )

        self.dense_layers = nn.Sequential(
            nn.Dropout(0.4),
            nn.Linear(50176, 1024),
            nn.ReLU(),
            nn.Dropout(0.4),
            nn.Linear(1024, K),
        )
```

```
def single_prediction(image_path):
    image = Image.open(image_path)
    image = image.resize((224, 224))
    input_data = TF.to_tensor(image)
    input_data = input_data.view((-1, 3, 224, 224))
    output = model(input_data)
    output = output.detach().numpy()
    index = np.argmax(output)
    print("Original : ", image_path[12:-4])
    pred_csv = data["disease_name"][index]
    print(pred_csv)
```

```
single_prediction("test_images/Apple_scab.JPG")
```

```
Original :  Apple_scab
Apple : Scab

<IPython.core.display.Javascript object>
```

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 3, March 2022

DOI: 10.17148/IJARCCE.2022.11326

CONCLUSION

Crop protection in organic farming is a difficult task. This is dependent on a thorough understanding of the crop being grown as well as potential pests, pathogens, and weeds. In our system, a special deep learning model based on a special architectural convolution network has been developed to detect plant diseases and their supplements using images of healthy or diseased plant leaves. The above-mentioned system can be upgraded to a real-time video entry system that enables unattended plant care. An intelligent system that Cures identified ailments is another feature that can be added to certain systems. According to research, managing plant diseases can help increase yields by up to 50%.

The purpose of this review was to explain DL approaches for detecting plant diseases. Furthermore, numerous visualisation techniques/mappings for recognising disease symptoms were summarised. Despite significant progress over the last three to four years, some research gaps remain, as described below:-

- The PlantVillage dataset was used to evaluate the accuracy and performance of the respective DL models/architectures in the majority of the studies (as described in the preceding sections). Despite the fact that this dataset contains a large number of images of various plant species and their diseases, it has a simple/plain background. However, for a realistic scenario, the real environment should be taken into account.
- Hyperspectral/multispectral imaging is a new technology that has been used in a variety of fields of study. As a result, it should be combined with efficient DL architectures to detect plant diseases even before symptoms appear.
- A more efficient method of visualising disease spots in plants should be implemented in order to save money by avoiding the use of fungicide/pesticide/herbicide that is unnecessary.
- Because the severity of plant diseases varies over time, DL models should be improved/modified to detect and classify diseases throughout their entire life cycle.
- The DL model/architecture should be efficient for a wide range of lighting conditions, so the datasets should not only represent the real environment but also include images taken in various field scenarios.
- To understand the factors influencing plant disease detection, such as dataset classes and size, learning rate, illumination, and so on, a comprehensive study is required.

REFERENCES

1. S. D. Khirade and A. B. Patil, "Plant Disease Detection Using Image Processing," 2015 International Conference on Computing
Communication Control and Automation, 2015, pp. 768-771, doi: 10.1109/ICCUBEA.2015.153.
2. S. C. Madiwalar and M. V. Wyawahare, "Plant disease identification: A comparative study," 2017 International Conference on Data
Management, Analytics and Innovation (ICDMAI), 2017, pp. 13-18, doi: 10.1109/ICDMAI.2017.8073478.
3. P. Moghadam, D. Ward, E. Goan, S. Jayawardena, P. Sikka and E. Hernandez, "Plant Disease Detection Using Hyperspectral
Imaging," 2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA), 2017, pp. 1-8, doi:
10.1109/DICTA.2017.8227476.
4. S. D.M., Akhilesh, S. A. Kumar, R. M.G. and P. C., "Image based Plant Disease Detection in Pomegranate Plant for Bacterial Blight,"
2019 International Conference on Communication and Signal Processing (ICCSP), 2019, pp. 0645-0649, doi: 10.1109/ICCSP.2019.8698007.
5. G. Shrestha, Deepsikha, M. Das and N. Dey, "Plant Disease Detection Using CNN," 2020 IEEE Applied Signal Processing
Conference (ASPCON), 2020, pp. 109-113, doi: 10.1109/ASPCON49795.2020.9276722.
6. Mohanty SP, Hughes DP and Salathé M (2016) Using Deep Learning for Image-Based Plant Disease Detection. Front. Plant Sci.
7:1419. doi: 10.3389/fpls.2016.01419
7. R. M. Haralick, K. Shanmugam and I. Dinstein, "Textural Features for Image Classification," in IEEE Transactions on Systems, Man,
and Cybernetics, vol. SMC-3, no. 6, pp. 610-621, Nov. 1973, doi: 10.1109/TSMC.1973.4309314.
8.Breiman,L.Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324.