DOI: 10.21917/ijsc.2023.0413

CLASSIFICATION OF PADDY LEAF DISEASES WITH EXTENDED HUBER LOSS FUNCTION USING CONVOLUTIONAL NEURAL NETWORKS

B. Sowmiya[1], K. Saminathan[2] and M. Chithra Devi[3]

[1,2]PG and Research Department of Computer Science, A.V.V.M. Sri Pushpam College, Affiliated to Bharathidasan University, India
[3]Department of Computer Science, Queens College of Arts and Science for Women, Affiliated to Bharathidasan University, India

Abstract

Paddy is a major food crop serving more than half the population of people in the world. It is inevitable to improve the quantity and quality of food crop with the growing population. Different factors including soil fertility, water availability, erratic climate variations, diseases, and pests, have an impact on paddy crop yield. It is crucial to identify the root cause for the reduction in yield of paddy. Early disease diagnosis prevents the plants from getting worst through its consecutive stage. The concern with manually diagnosing plant leaf diseases with the naked eye is that the results can be less accurate and even unreliable. Automatic disease diagnosis eliminates the need for experts and provides accurate results. This paper will assist the farmers to identify the leaf diseases automatically with the aid of Convolutional Neural Networks. This research includes paddy leaf disease categories: bacterial blight, blast, tungro, brown spot and healthy leaves. The dataset contains 800 images, 160 images from each of the five categories. Images are resized to 256 * 256 pixels and normalized. The network architecture created with convolutional, maxpooling, flatten and dense layers. The Dataset is divided into training and validation set in 70:30 ratios and model is trained with 20 epochs of batch size 16. The novelty of the study is the implementation of extended Huber loss function for minimizing the loss. Furthermore, it is cross compared with existing loss functions. The Proposed model has achieved 96.63% training accuracy and 86.61% validation accuracy with 5 classes. Performance of model is evaluated with confusion matrix with precision, recall, F1-score and support as parameters.

Keywords:

Paddy Disease Detection, Preprocessing, Classification, Huber Loss, Convolutional Neural Network

1. INTRODUCTION

In India, 54.6% of people are dependent mostly on agriculture for their living. The economy of a country depends on the agriculture. As per the report of Ministry of Agriculture and Farmers' Welfare, APEDA, Ministry of Commerce, rice crop turns around 6.89 billion US dollar for the FY 2022 (from April 2021 until December 2021). As per Union budget 2022-23, Rs.8,514 crore allocated to Department of Agricultural Research and Education to increase production of crops and soil enhancement [1]. Paddy is one among the more demanding food crop in South Asia. But the plant diseases decrease productivity of crop [2]. Agricultural productivity impacts the economy of our country. 18% of India's GDP is obtained from the agricultural sector. With the growing population, it is essential to increase productivity by preventing plant from diseases [3]. The health of food crop is essential for attaining crop security and sustainability in agriculture. But, due to variety of factors, plants are affected by diseases, which in turn will affect the quality and quantity of crop. Proper pesticides should be applied on the early stage of disease to avoid soil pollution. Plants are affected by pathogens, fungi, bacteria, viruses and other microbes. Early detection of diseases in plants will result in increase in quantity and quality of crop yield. By usual way, plants are detected for illnesses through manual identification through naked eyes. It needs expert consultation which is time consuming and expensive [4]. Diseases can be identified in all the parts of plant. Yet the symptoms are clearly visible in leaves because the leaves have plane nature and easy to interpret without any difficulties compared to other parts of plant [5]. Numerous improvements are made following the introduction of the green revolution in India. Pesticides and fertilizers are its consequences, have both positive and negative sides. Though plants are protected, soil gets polluted. So, it's vital to use pesticides without damaging crops [6].

As manual leaf disease has several drawbacks like time consumption, need of expert consultation, poor accuracy, etc., the technology assists in finding proper solutions. Through the tremendous advancements in the field of image processing with machine learning, deep learning and artificial intelligence, there have been solutions to most of the problems in agricultural sector. With the aid of these technologies, the farmer can detect plant diseases without the guidance of expert. So, automated process of leaf disease diagnosis is introduced to produce better results [7]. Early disease diagnosis is essential not only to improve crop yield and quality of crop but also to prevent the plants from diseases [8]. Now days CNN has gained more attention than traditional machine learning due to its automatic feature extraction which is time consuming, and CNN provides end-to-end learning. Convolutional Neural Networks is also found to perform better than Neural Networks [9].

The objective of study is to develop an efficient automated system with convolutional neural networks that can diagnose diseases like bacterial blight, blast, brown spot, tungro and healthy leaf, with fast and accurate result at least expense.

The rest of the paper is organized as follows. Section 2 explores the existing work done by various researchers with tools and techniques. Section 3 presents the implementation of proposed system with the detailed explanation of proposed CNN architecture utilized. Section 4 comprises discussion of results with enough visualization. Finally, section 5 concludes the paper with the futuristic work.

2. LITERATURE REVIEW

The research work of same problem that has already been done is elaborated in this part. Researchers have developed a range of methods for computer vision-based plant leaf disease diagnostics. The literature heavily relies on CNN.

In a study by Sharma et al. [2], a CNN model is created to predict and classify diseases in paddy plant. As bacterial, viral and fungal are three disease families, the authors considered brown spot, leaf smut and bacterial leaf blight. Training samples pass

under convolution layer, max pooling layer, flatten layer, dense layer, fully connected layer. Data augmentation like horizontal flip, vertical flip, shearing, and brightening of images is carried out on dataset, which is self-collected. Test samples are loaded. Data is fit and model is trained on training samples. Accuracy of model is predicted with test dataset. An accuracy of 90.32% is achieved in test set and 93.58% is achieved the training set.

Shrestha et al. proposed CNN based model for disease detection in a variety of crops, as CNN approaches regularization in a simpler fashion. Dataset is loaded and are converted to numpy array. Data is labeled and split into training and testing sets. Model is built and trained and tested on training and testing set. Disease is predicted. Filters acquire the ability to recognize abstract concepts like a person face boundary. Instead of being predetermined, they are self-learned. The results of the filter are feature maps. Input image is passed into convolution layer that extracts characteristics and maintains in a feature map. Feature map is passed to pooling layer, where 2D filter applied and this layer summarizes the feature present in a region of feature map. The most prominent features of previous layer will be the output for next layer, max pooling layer. Fully connected layer is a feed forward neural network, which contains the vital information from all the layers. With 3000 images with 15 classes, model shows training accuracy of 97.2% and test accuracy of 88.80% and not overfitted [3].

Tejaswini et al. have considered frequent rice diseases: brown spot, hispa, leaf blast along with healthy leaves. Different DL methods like VGG-16, VGG-19, ResNet, Xception, 5-layer CNN. VGG-16 is a 16-layer CNN. VGG-19 is a 19-layer CNN. ResNet solves the vanishing gradient decent problem. Xception has 71 layers. Convolution, pooling and fully connected layers is the layers involved. 5-layer CNN has two additional layers: dropout and activation layers. Among variant DL modes, 5-layer CNN outperforms with 78.2% accuracy, whereas VGG-16 show low accuracy of 58.4% [4]. Saini et al. proposes deep learning CNN model using 5000 leaf images of apple and grapes utilizing convolution layer, maxpooling layer, flatten layer, dense layer. The best training and testing accuracies are reported to be 99.96% and 99.90%. They assert that manual feature extraction is no longer necessary with deep learning, and the key challenge is the requirement for a high-speed GPU for data processing [5].

Swathika et al. proposes a CNN model to detect leaf disease in paddy. Image is acquired from Kaggle dataset and resized to 500*100 pixels. Dataset is divided into 90:10 ratios for training and testing and sent to neural network. Disease identified images are sent to contour detection model and affected leaf area is estimated. Binary and Otsu's thresholding is done in contour module. The accuracy and value accuracy are observed to be approximately 70% [6]. Islam et al. proposes deep CNN model to predict and classify paddy disease categories namely: brown spot, leaf blast, leaf blight, leaf smut and healthy leaf. Dataset is collected from UCI machine learning repository and Kaggle. Image is subjected to preprocessing like resizing, rotation, zoom and shearing. VGG-19, ResNet-101, Xception and Inception-ResNet-V2 architectures are used. Inception-ResNet-V2 outperforms with 92.86% accuracy. Transfer learning is adapted which is shown to increase accuracy and reduce complexity of model's training time. Usage of Pre-trained weights in a new model can perform well than general models. Transfer learning helps to reuse previously learned model on a new model. Also, it can train deep neural networks with small amount of data [7].

In a recent study, Bari et al. proposes real time disease detection using Faster R-CNN algorithm where challenges faced by existing work like difficulty in diseased are segmentation with small variations, overfitting problem caused by discrepancies in distribution of data features, large variety of disease feature, complex backgrounds, and obscure boundaries of disease symptoms are discussed. YOLO, SSD and Faster R-CNN algorithms are considered. Among them Faster R-CNN proposes RPN (Regional Proposal Network) structure to generate candid regions and target is precisely positioned. R-CNN efficiently detects spot of disease reliably. Data augmentation is done to avoid overfitting issue. The faster R-CNN network effectively classifies rice diseases with high accuracy in real time. Three diseases including rice blast, brown spot, hispa and are the categories handled along with healthy one. Rice leaf diseases dataset (RLDD) is created with online and own dataset collected from rice fields. RLDD is annotated manually, and data augmentation done. Entire dataset is divided into training and testing dataset. Testing dataset is used for performance assessment. Model has batch size of 1 with 50,965 iterations and 0.0002 as learning rate. Rice blast, brown spot, hispa and healthy leaf classes accuracy of 98.09%, 98.85%, 99.17% and 99.25% respectively with faster R-CNN. The study reports that the misclassification is due to similarity in geometric features and to combat this problem, more training should be done with more similar dataset [8].

Shrivastava et al. proposed a decision support system for identifying rice plant disease using SVM and ANN (Artificial Neural Network) where ANN is reported to be accurate than SVM [10]. Nayak et al. proposed a system using improved deep CNN to recognize facial expression with adabound optimizer [11]. Janocha et al. investigates how the choice of loss functions influencing deep models, learning dynamics and consequential classifiers robustness to various effects [12]. Latif et al. proposed deep CNN transfer learning method for diagnosing six categories of rice leaf diseases and achieved 96.08% accuracy and tacked overfitting issue [13]. Thakur et al. presents a lightweight CNN model for crop disease identification, called VGG-ICNN, based on the VGG architecture. It reduces the size of the model while maintaining 99.16% accuracy and outperforms other lightweight CNN models for crop disease identification [14]. Prottasha et al. proposes a system to identify 12 types of rice leaf diseases using depth wise separable CNN, which deals not only with spatial dimension but also with depth dimension. Model reports 96.5% validation and 95.3% testing accuracy [15]. Hassan et al. proposed a method for identifying plant-leaf diseases by fine-tuning a pre-trained Convolutional Neural Network (CNN) on a dataset of plant leaves with annotated disease labels. The pre-trained CNN serves as the base model, with its features used to extract information from the images, while the fine-tuning process focuses on training the last layers of the network to perform the specific task of plant-leaf disease classification. Experiments on a dataset of plant leaves showed that this approach outperforms traditional machine learning methods and achieves high accuracy in plant-leaf disease classification. The paper concludes that transfer learning is a promising approach for this type of classification and has potential to be applied to other similar

problems in computer vision. Highest accuracy of 99.56% is achieved using EfficientNetB0 [16].

Geethamani et al. proposed nine-layer Deep CNN model for identifying plant leaf diseases. Extensive augmentations performed. Model evaluated with varying epoch, batch size and dropouts and achieved 96.46% classification accuracy. They have tested the proposed model with machine learning algorithms as well. The future work is to extend the research to other parts of plant like stem, flowers and fruits [17].

In this research, a study on the detection of leaf diseases using a hybrid convolutional neural network (HCNN) is presented. By using feature reduction approaches, the authors hope to enhance the model's performance. A dataset of images of plant leaves is used to train and test the HCNN model, and the results are assessed using a variety of performance criteria. The study's methodology and findings are reported in the paper, along with how feature reduction affects the HCNN model's ability to identify leaf diseases accurately and effectively. Four diseases in grape plants namely: Leaf blight, Black rot, stable, and Black measles are investigated and accuracy of 98.7% is achieved using transfer learning and efficientNetB7 [18]. This research proposes a different probabilistic interpretation of the Huber loss, an often-used loss function in regression problems. They provide a probabilistic interpretation of the Huber loss function in terms of Gaussian mixture models. The paper discusses the objectives, methods, and results of the study and compares the Huber loss performance with other commonly used loss functions in regression problems [19].

3. PROPOSED WORKFLOW

The proposed system intends to create an effective plant disease detection mechanism for paddy plants using combination of Convolutional Neural Networks and image processing techniques. This section provides a thorough explanation of the proposed system. The Fig.1 shows complete workflow of proposed system. The images are acquired from P.K. Sethy [20] repository. The image dataset is preprocessed by reducing to 256*256 dimensions. All pixel values will result in 0-255 range, which is then normalized by dividing each image pixel with 255. The images are acquired from P.K. Sethy [20] repository and Kaggle dataset [21].

3.1 IMAGE ACQUISITION

The first and foremost step is image acquisition. Images are acquired from dataset of P.K. Sethy [20] having 5932 images in their repository of which 640 images (160 image samples from each of the four classes of paddy leaf diseases: bacterial blight, blast, brown spot, and tungro) are acquired. 160 healthy images are taken from Kaggle dataset [21]. The Fig.2 depicts sample images of diseased and healthy leaf.

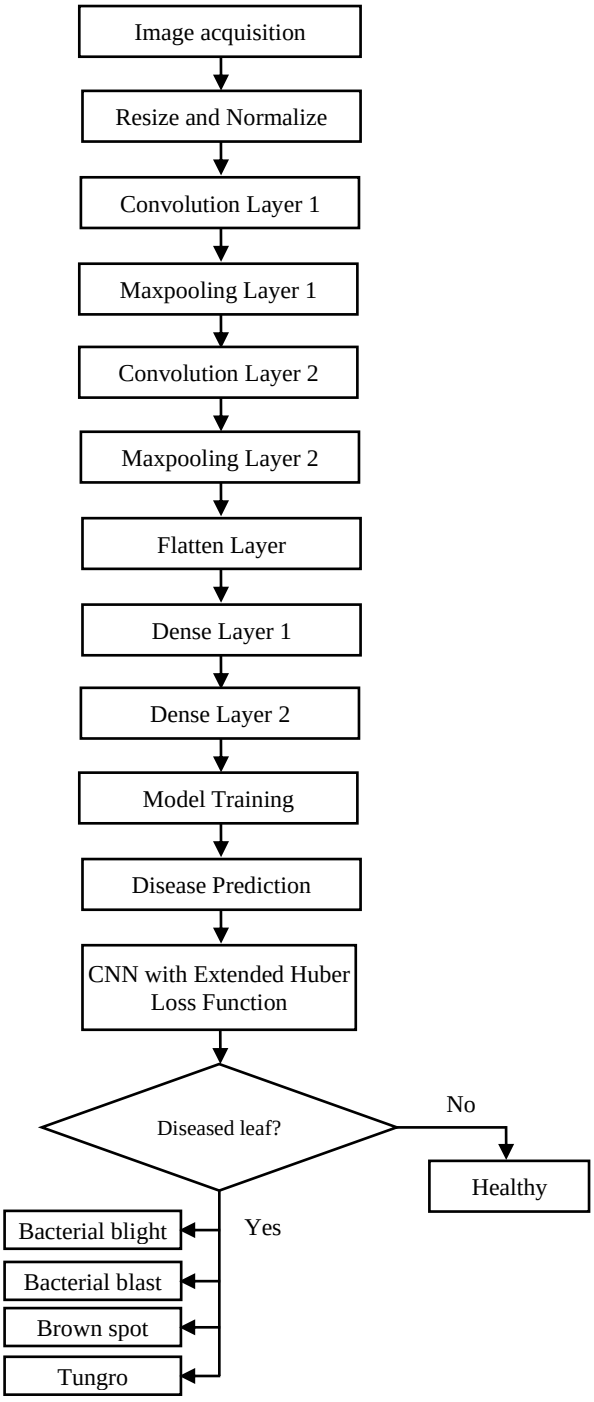

Fig.1. Proposed system workflow

(a) bacterial blight (b) blast (c) brown spot

(d) tungro (e) healthy leaf

Fig.2. Sample paddy leaf diseases and healthy leaf

3.2 PREPROCESSING

Preprocessing improves the quality of image. We have resized all the images into 256*256 pixels. Python's CV2 package is used to resize the images and images are converted into numpy array. Dataset is scaled to a range of 0 to 1 before it is fed into the neural network model. It is done by divide the values by 255. Both the training set and the validation set should be preprocessed in the same method. Table.1. depicts the dataset information used in this research.

Table.1. Total dataset of classes

Class Name	Total Images	Total Training Images	Total Testing Images
Brown Spot	160	128	32
Leaf Blast	160	128	32
Leaf blight	160	128	32
Leaf smut	160	128	32
Healthy	160	128	32
Total images	800	640	160

3.3 CNN ARCHITECTURE

CNN architecture is utilized for image classification, object recognition, segmentation and many other tasks. CNN takes image as input and interprets image in the form of matrices. CNN Architecture has three main layers namely convolutional layer, pooling layer and fully connected layer. CNN Layers are discussed below.

3.3.1 Proposed CNN model:

We propose a CNN model with two convolutional layers, a dropout layer, two pooling layers, a flatten layer and two dense layers. The process is explained below and depicted in Fig.3.

Convolutional layer is added where convolution operation is applied on input image with 32 filters and 4*4 filter size. This step extracts useful features and ReLU activation function is internally applied.

Step 1: To reduce the dimensionality of images, a max-pooling layer with pool-size 4*4 is added.

Step 2: A second-convolutional layer with 16 filters and 4*4 filter size is applied with ReLU as activation function.

Step 3: A second max-pooling layer of 4*4 pool-sizes is added.

Step 4: A dropout layer with 0.2 dropout rate is added to drop off some neurons [11].

Step 5: Then a flattening layer is applied to convert two dimensional arrays obtained from pooled feature maps into one dimensional linear vector.

Step 6: Finally, two dense layers are applied, first layer employs ReLU and second layer employs softmax activation function.

Conv2d_input	input:	[(None, 256, 256, 3)]
InputLayer	output:	[(None, 256, 256, 3)]
↓		
Conv2d	input:	(None, 256, 256, 3)
Conv2D	output:	(None, 256, 256, 32)
↓		
Max_pooling2d	input:	(None, 256, 256, 32)
MaxPooling2D	output:	(None, 64, 64, 32)
↓		
Conv2d_1	input:	(None, 64, 64, 32)
Conv2D	output:	(None, 64, 64, 16)
↓		
Max_pooling2d_1	input:	(None, 64, 64, 16)
MaxPooling2D	output:	(None, 32, 32, 16)
↓		
dropout	input:	(None, 32, 32, 16)
Dropout	output:	(None, 32, 32, 16)
↓		
flatten	input:	(None, 32, 32, 16)
Flatten	output:	(None, 16384)
↓		
dense	input:	(None, 16384)
Dense	output:	(None, 10)
↓		
dense_1	input:	(None, 10)
Dense	output:	(None, 5)

Fig.3. Proposed CNN model with input and output of layers

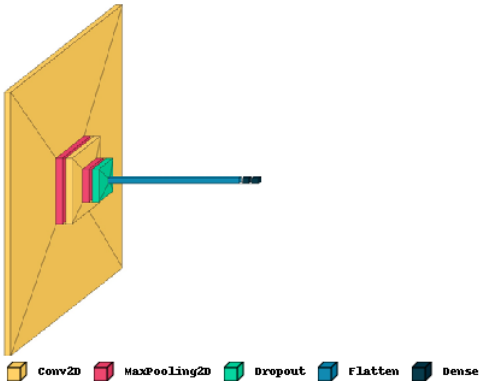

Fig.4. Layered view of CNN model

The image passes through all the layers and classification is finally done in dense layer based on inputs from previous layers

and model will be able to predict the disease class to which the image belongs to. The Fig.4 shows the layered view of proposed CNN model where different CNN layers are involved.

3.3.2 Convolution layer:

This is the first layer of CNN which involves more computations. It requires input data that has height, width, and depth as dimensions. Each filter will result in feature map because of convolution operation with input image. Filter is applied to input image and returns "feature map", because of convolution operation with input image, which is then normalized (with an activation function) and/or resized. This process can be repeated several times. Resulting feature map is W×H×D, where W is its width in pixels, H is its height in pixels and D the number of channels (It is 1 for a black and white image and 3 for a color image). Finally, the values of the last feature maps are concatenated into a vector. This vector defines the output of the first block and the input of the second. The depth of output is affected by the number of filters, or they define the dimensionality of the output space. The coding for adding convolutional layer is as follows.

```
model.add (Conv2D (32,(4,4), padding = "same", input_shape =
                (256,256,3), activation = "relu"))
```

ReLU (Rectified Linear Unit) is a popular activation function and it is simple to compute as it uses only max() function. It is defined as follows.

$$f(x) = \max(0,x) \qquad (1)$$

where, x is the input neuron.

3.3.3 Maxpooling layer:

Pooling layers reduce the number of parameters in the input known as dimensionality reduction which decreases the computational power to process the data and dominant features are extracted as well. The operation of pooling layer is similar to convolution layer, where the filter is applied to entire image, but the filters carry no weight.

By factoring in the maximum value over an input window for each channel of input, this layer reduces the input's spatial dimensions: height and width. This layer will result in pooled feature map. The code to add pooling layer is as follows.

```
model.add(MaxPooling2D(pool_size=(4, 4)))
model.add(MaxPooling2D(pool_size=(2,2)))
```

3.3.4 Flatten:

The pooling layer produces feature map, which is passed through flatten layer, where everything is flattened into a long column that is transformed into one-dimensional vector. This vector will be input to the neural network. Fig.5 represents pooled feature map and results after flattening.

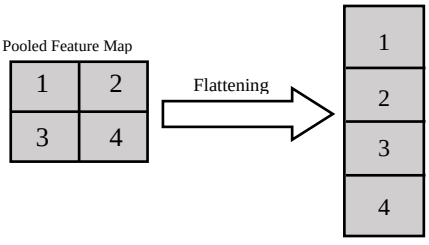

Fig.5. flattening of pooled feature map

The code to add flatten layer is as follows: model.add(Flatten())

3.3.5 Dense Layer:

Dense layer is the layer having highest number of parameters. This layer performs some linear operations like matrix-vector multiplication with neurons of its preceding layer and providing one output to next layer, hence this layer is a fully connected layer.

ReLU and softmax activation functions are used for the dense layer. ReLU activation function helps the model to learn complex patterns in data by introducing non-linearity into model. Softmax function is used for multiclass classification problem. The code to add dense layers is as follows.

```
model.add(Dense(8, activation="relu"))
model.add(Dense(4, activation="softmax"))
```

Eq.(2) and Eq.(3) displays the formula for ReLU and Softmax activation function.

$$ReLU(x) = \max(0, x) \qquad (2)$$

where, x is the input to the activation function and f(x) is the output.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \qquad (3)$$

where z is the input vector, e^{z_i} is standard exponential function, k is number of classes in multiclass classifier.

- ***Forward Propagation***

Each layer's output in forward propagation is determined by its input and weight. The following are the steps for calculating each layer's output:

Step 1: The input of the current layer and the weights for that layer is dot product.

Step 2: The bias term is added to result of dot product.

Step 3: The result is passed to get output of current layer through activation function.

Eq.(4a) and Eq.(4b) denotes intermediate steps and output of current layer.

$$z = input * weights + bias \qquad (4a)$$

$$output = f(z) \qquad (4b)$$

- ***Backward Propagation***

The gradient of loss function is calculated with respect to each weight. The steps for backpropagation are as follows:

Step 1: The input is forward propagated through the network to calculate the output.

Step 2: The difference between the desired output and the actual output is computed.

Step 3: The gradient of the error with respect to the output of the last layer is calculated.

Step 4: Backpropagate the error through the network, calculating the gradient of the error with respect to the weights in each layer.

Step 5: This process is repeated for each layer and each weight in the network.

The Eq.(5) shows formula for gradient of the error.

$$error\_grad = (output - desired\_output) * output\_grad \quad (5)$$

where, *error_grad* is the gradient of the error with respect to the output of the last layer, output is the actual output of the network, *desired_output* is the desired output and *output_grad* is the derivative of the output with respect to the input. It is also known as the activation function gradient.

The gradient of the error with respect to the weights in each layer is then calculated using the chain rule as in Eq.(6) as follows:

$$weight\_grad = input * error\_grad \quad (6)$$

where *weight_grad* is the gradient of the error with respect to the weights in the current layer, input is the input to the current layer, and *error_grad* is the gradient of the error with respect to the output of the current layer. The *weight_grad* is used to update the weights using the gradient descent algorithm and is shown in Eq.(7).

$$weights = weights - learning\_rate * weight\_grad \quad (7)$$

3.4 CLASSIFICATION

Classification involves classifying data into different classes. This research involves multiclass CNN classification, where CNN is trained on a dataset and then used to predict the class. Dataset is divided into train and test in 70:30 ratios. Training a network is trying to minimize its loss. The loss function defines the difference between the predicted class and true class. It guides the optimization process by assigning a cost to model's prediction error. The proper choice of loss function for a classification problem has a great impact on the model's performance, convergence speed, and overall accuracy.

Steps for classification:

Step 1: Build the model.

Step 2: Set up the required layers.

Step 3: Compile the model with Extended Huber loss function, optimizer and metrics.

Step 4: Train the model.

Step 5: Feed the model.

Step 6: Evaluate accuracy.

Step 7: Make predictions and verify predictions.

Step 8: Use the trained model to make predictions.

3.4.1 Proposed Model with Extended Huber Loss Function:

Huber loss is a loss function used in robust regression [19]. It is less sensitive to outliers in the data than the mean squared error used in least squares. They produce more stable and reliable results. The function is a combination of the mean squared error for small errors and mean absolute error for large errors. Huber loss function is defined in Eq.(8a) and Eq.(8b).

$$L(x) = 0.5 * (error^2), \text{ if } |error| \leq \delta \quad (8a)$$

$$L(x) = \delta * |error| - 0.5 * (\delta), \text{ if } |error| > \delta \quad (8b)$$

where error is the difference between the predicted value and the true value, and δ (delta) is a parameter that controls the transition between the two regimes, "linear" and "quadratic" regions of the loss function.

- ***Extended Huber loss function***

In order to improve the results further, an improved version of the Huber loss function is created, which is less sensitive to the choice of the delta parameter, and more robust to outliers and has better optimization properties. Improved Huber loss has a smooth transition between the quadratic and linear parts of the loss function, whereas the standard Huber loss has a sharp transition.

The extended Huber loss function is defined as:

$$L_1(x) = 0.5 * (error_1^2), \text{ if } |error_1| \leq \delta \quad (9a)$$

$$L_1(x) = \delta * |error_1| - 0.5 * (\delta^2), \text{ if } |error_1| > \delta \quad (9b)$$

where, $error_1$ is the difference between the predicted value and the true value, and δ (delta) is a hyperparameter that controls the transition from the quadratic to linear part of the loss function. Extended Huber loss function produces better results in terms of lower loss and higher accuracy than the standard Huber loss function. It has better optimization properties and less prone to oscillation. It has a smooth transition between the quadratic and linear parts of the loss function, which makes it less sensitive to outliers. It is less sensitive to the choice of the delta parameter, which makes it more robust to outliers. Moreover, it is differentiable, enabling the use of gradient-based optimization techniques.

4. RESULTS AND DISCUSSION

The Table.2 depicts detailed architecture with the layers, no. of neurons in each layer and no. of parameters.

Table.2. Detailed architecture of proposed CNN model

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 256, 256, 32)	1568
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_1 (Conv2D)	(None, 64, 64, 16)	8208
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 16)	0
dropout (Dropout)	(None, 32, 32, 16)	0
flatten (Flatten)	(None, 16384)	0
dense(Dense)	(None,10)	163850
dense_1(Dense)	(None,5)	55
Total params: 173,681		
Trainable params: 173,681		
Non-trainable params: 0		

4.1 VISUALIZATION OF FEATURE MAP

The Fig.6 shows feature map visualization obtained through convolutional layers.

With equal distribution of data in five classes, our proposed model has a total of 800 images with 640 images for training and 160 images for testing. The python code was executed for several rounds of epochs to track resulting accuracy and losses.

From epoch 1 to 6, gradual increase is noticed and slight decline at epoch 7. From epoch 8 to 20, model is showing steady increase rate. The best training accuracy of 96.63% is reached at the 20th epoch. The study reports that there is gradual increase in validation accuracy from epoch 1 to 5 and slight decline noticed in epoch 6, 8, 14, 17 and 19. The highest validation accuracy of

86.61% is attained in epoch 12, 13 and 16. There is a gradual decline in training loss in all epochs except epoch 18, which shows a slight increment. There is a gradual decrease in validation loss in epoch 3, 8, 10, 13, 17 and 19.

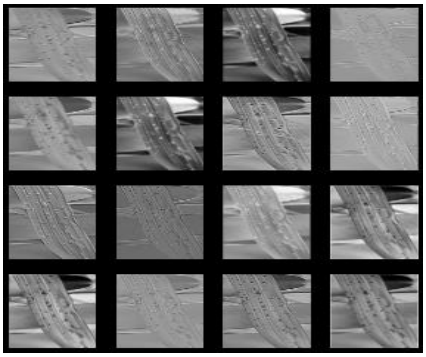

Fig.6. Feature map visualization

The Table.3 displays the results of the proposed model's loss, accuracy, and validation loss and validation accuracy throughout several epochs.

Table.3. Full model training details with extended Huber loss function

Epoch	Training		Validation	
	Loss	Acc.	loss	Acc.
1	0.0590	0.4442	0.0572	0.4286
2	0.0512	0.5603	0.0461	0.5982
3	0.0434	0.6384	0.0469	0.5983
4	0.0389	0.7143	0.0381	0.7232
5	0.0340	0.7612	0.0336	0.8036
6	0.0302	0.8304	0.0307	0.7946
7	0.0301	0.8259	0.0302	0.8393
8	0.0247	0.8661	0.0341	0.7589
9	0.0223	0.8817	0.0258	0.8393
10	0.0205	0.8884	0.0269	0.8482
11	0.0181	0.8996	0.0267	0.8571
12	0.0168	0.9152	0.0230	0.8661
13	0.0139	0.9308	0.0237	0.8661
14	0.0133	0.9308	0.0250	0.8482
15	0.0114	0.9464	0.0245	0.8571
16	0.0110	0.9487	0.0208	0.8661
17	0.0085	0.9576	0.0276	0.8036
18	0.0086	0.9554	0.0188	0.8571
19	0.0074	0.9598	0.0268	0.8304
20	0.0068	0.9665	0.0203	0.8482

The Fig.7 shows the loss and accuracy curve of the proposed CNN model for 20 epochs. Training and validation loss have similar trends, though they differ in absolute values. Proposed model is cross compared with standard Huber loss function and categorical cross entropy loss functions and it is emphasized that overfitting is under control with an extended Huber loss function.

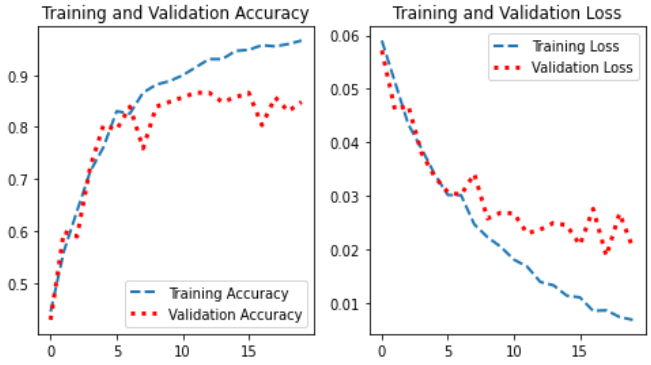

Fig.7. Extended Huber loss function – loss and accuracy plot

Table.4 shows that extended Huber loss function outperforms the standard Huber loss and standard loss function like categorical cross entropy.

Table.4. Comparison of loss functions w.r.t loss and accuracy

Loss function	Training		Validation	
	Loss	Acc.	Loss	Acc.
Categorical cross entropy	32.32	91.74	55.48	84.82
Standard Huber loss	0.80	95.31	2.51	83.93
Extended Huber loss	**0.68**	**96.63**	**2.03**	**86.61**

The Fig.8 depicts loss and accuracy plot with standard Huber loss function where overfitting is under control.

The Fig.9 depicts loss and accuracy plot with categorical cross entropy. The model shows 91.74% training accuracy with higher training and accuracy loss ranges than standard and extended Huber loss functions. The Fig.10 and Fig.11 shows training and testing loss for standard and extended Huber loss functions.

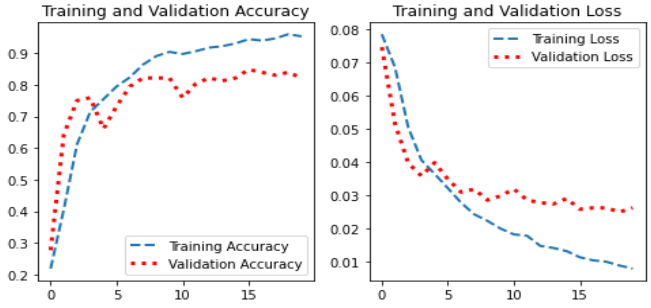

Fig.8. Standard Huber loss function – loss and accuracy plot

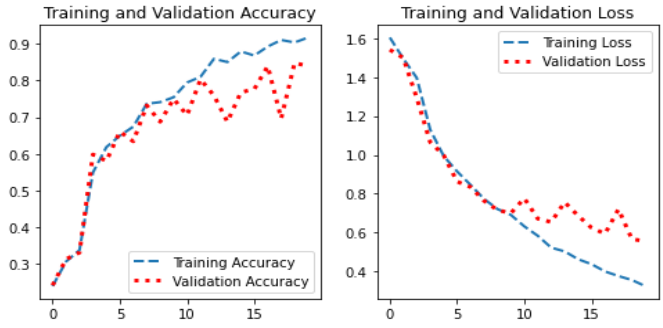

Fig.9. Categorical cross entropy - loss and accuracy plot

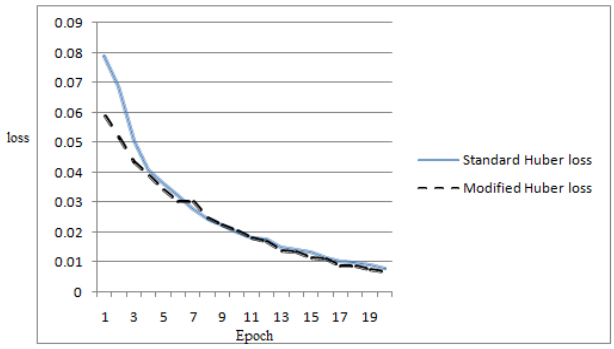

Fig.10. Training Loss - standard Vs extended Huber loss

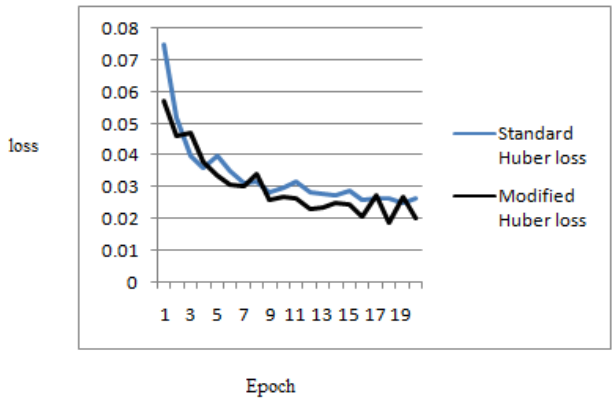

Fig.11. Validation loss - standard Vs extended Huber loss

The Table.5 shows the comparison of accuracy of proposed system with the existing systems.

Table.5. Comparison of Proposed system with existing systems

References	Accuracy
Sharma et al. [1]	93.58%
Tejaswini et al. [3]	78.20%
Swathika et al. [5]	70%
Sibiya et al. [8]	92.85%
Proposed CNN extended Huber loss	96.65%

Confusion matrix evaluates the performance of classification model by comparing actual target values with the predicted values. In our paper, confusion matrix incorporates performance evaluation metrics like precision, recall, support, and F1-score to assess the efficiency of the classification system. Confusion matrix estimates are explained in Table.6.

Table.6. Key terms in Confusion matrix

Outcome	Explanation
TP	True Positive, shows correctly predicted positive values
TN	True negative, shows correctly predicted negative values
FP	False Positive, shows actual negatives that are incorrectly predicted
FN	False Negative, shows incorrect prediction of negative values

Formula for precision, recall and F1-score are given in Eq.(10), Eq.(11), and Eq.(12).

$$\text{Recall} = TP/(TP+FN) \quad (10)$$

$$\text{Precision} = TP/(TP+FP) \quad (11)$$

$$\text{f1-score} = 2*((precision*recall)/(precision + recall)) \quad (12)$$

For the proposed system, an Intel(R) Core (TM) i3-5005U CPU with 4GB of RAM and a 64-bit processor is employed. Anaconda 1.9.0 and Python 3.9.7 are the software versions used.

The Fig.13 displays confusion matrix for the proposed method where the diagonal element represents correctly classified count. It is reported from Fig.13 that 156 images are correctly classified as bacterial blight.

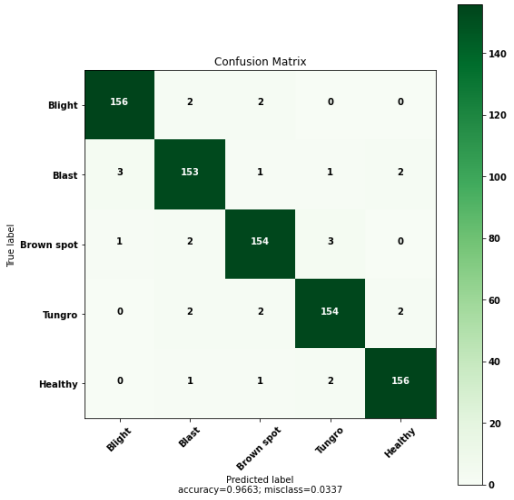

Fig.13. Confusion matrix for proposed system

The Table.7 shows classification report for the proposed model with precision, recall, F1-score and support parameters.

Table.7. Classification report of proposed CNN

	Precision	Recall	F1-score	Support
Bacterial blight	97.50	97.50	97.50	160
Blast	95.62	95.62	95.62	160
Brown spot	96.25	96.25	96.25	160
Tungro	96.25	96.25	96.25	160
Healthy	97.50	97.50	97.50	160
Accuracy			96.63	800
Macro avg	96.63	96.63	96.63	800
weighted avg	96.62	96.62	96.62	800

5. CONCLUSION

Plant disease diagnosis is a crucial area to work with. Automatic plant disease diagnosis helps the farmers to increase yield of product and to protect soil from improper use of pesticides. This study attempts to diagnose and classify paddy leaf diseases using deep learning-based CNN approach. The proposed work achieves training accuracy of 96.63% and validation accuracy of 86.61% having considered four classes of diseases: bacterial blight, blast, tungro and brown spot and healthy leaf. The

main advantage of deep learning methods over machine learning methods is that they eliminate the need for feature extraction. The futuristic work of this paper is to add on more classes of diseases and to implement with the variants of deep learning architecture for sustainable computing.

REFERENCES

[1] Agriculture and Allied Industries, Available at https://www.ibef.org/download/1658816319_Agriculture-and-Allied-Industries-June-2022.pdf, Accessed in 2023.-

[2] R. Sharma and M. Pandey, "A Model for Prediction of Paddy Crop Disease using CNN", *Proceedings of International Conference on Progress in Computing, Analytics and Networking*, pp. 533-543, 2020.

[3] G. Shrestha and N. Dey, "Plant Disease Detection using CNN", *Proceedings of International Conference on Applied Signal Processing*, pp. 109-113, 2020.

[4] P. Tejaswini, Y.K. Rathore and R.R. Janghel, "Rice Leaf Disease Classification using CNN", *Proceedings of International Conference on Earth and Environmental Science*, pp. 12017-12023, 2022.

[5] G. Saini and A.K. Luhach, "Classification of Plants using Convolutional Neural Network", *Proceedings of International Conference on Sustainable Technologies for Computational Intelligence*, pp. 547-558, 2020.

[6] R. Swathika and K. Sowmya, "Disease Identification in Paddy Leaves using CNN based Deep Learning", *Proceedings of International Conference on Intelligent Communication Technologies and Virtual Mobile Networks*, pp. 1004-1008, 2021.

[7] M.A. Islam and T. Khatun, "An Automated Convolutional Neural Network based Approach for Paddy Leaf Disease Detection", *International Journal of Advanced Computer Science and Applications*, Vol. 12, No. 1, pp. 1-13, 2021.

[8] B.S. Bari, A.F. Ab Nasir and M. Majeed, "A Real-Time Approach of Diagnosing Rice Leaf Disease using Deep Learning-based Faster R-CNN Framework", *Peer Journal on Computer Science*, Vol. 7, pp. 432-443, 2021.

[9] M. Sibiya and M. Sumbwanyambe, "A Computational Procedure for the Recognition and Classification of Maize Leaf Diseases Out of Healthy Leaves using Convolutional Neural Networks", *AgriEngineering*, Vol. 1, No. 1, pp. 119-131, 2019.

[10] G. Shrivastava and H. Patidar, "Rice Plant Disease Identification Decision Support Model using Machine Learning", *ICTACT Journal on Soft Computing*, Vol. 12, No. 3, pp. 2619-2627, 2022.

[11] H.D. Nayak and A.K. Sarvaiya, "Facial Expression Recognition based on Feature Enhancement and Improved Alexnet", *ICTACT Journal on Soft Computing*, Vol. 12, No. 3, pp. 2589-2600, 2022.

[12] K. Janocha and W.M. Czarnecki, "On Loss Functions for Deep Neural Networks in Classification", *Proceedings of International Conference on Progress in Computing and Analytics*, pp. 1-7, 2022.

[13] G. Latif and Z.A. Kazimi, "Deep Learning Utilization in Agriculture: Detection of Rice Plant Diseases using an Improved CNN Model", *Plants*, Vol. 11, No. 17, pp. 2230-2243, 2022.

[14] P.S. Thakur and A. Ojha, "VGG-ICNN: A Lightweight CNN Model for Crop Disease Identification", *Multimedia Tools and Applications*, Vol. 87, pp. 1-24, 2022.

[15] S.I. Prottasha and S.M.S. Reza, "A Classification Model based on Depthwise Separable Convolutional Neural Network to Identify Rice Plant Diseases", *International Journal of Electrical and Computer Engineering*, Vol. 12, No. 4, pp. 1-12, 2022.

[16] S.M. Hassan and E. Jasinska, "Identification of Plant-Leaf Diseases using CNN and Transfer-Learning Approach", *Electronics*, Vol. 10, No. 12, pp. 1388-1398, 2021.

[17] G. Geetharamani and A. Pandian, "Identification of Plant Leaf Diseases using a Nine-Layer Deep Convolutional Neural Network", *Computers and Electrical Engineering*, Vol. 76, pp. 323-338, 2019.

[18] P. Kaur and A.M. Alabdali, "Recognition of Leaf Disease using Hybrid Convolutional Neural Network by Applying Feature Reduction", *Sensors*, Vol. 22, No. 2, pp. 575-584, 2022.

[19] G.P. Meyer, "An Alternative Probabilistic Interpretation of the Huber Loss", *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 5261-5269, 2021.

[20] Mendeley Data, "Rice Leaf Disease Image Samples", Available at https://www.kaggle.com/datasets/minhhuy2810/rice-diseases-image-dataset, Accessed in 2021.