Electronic Research Archive

http://www.aimspress.com/journal/ERA

ERA, 31(5): 2813–2833.
DOI: 10.3934/era.2023142
Received: 15 December 2022
Revised: 09 February 2023
Accepted: 10 February 2023
Published: 14 March 2023

Research article

Detection of rice plant disease from RGB and grayscale images using an LW17 deep learning model

Yogesh Kumar Rathore[1], Rekh Ram Janghel[1], Chetan Swarup[2,*], Saroj Kumar Pandey[3], Ankit Kumar[3,*], Kamred Udham Singh[4,*], and Teekam Singh[5]

[1] Department of Computer Science & Engineering, National Institute of Technology, Raipur, India
[2] Department of Basic Science, College of Science and Theoretical Studies, Saudi Electronic University, Riyadh-Male Campus 13316, Saudi Arabia
[3] Department of Computer Engineering & Application, GLA University Mathura, UP, India
[4] School of Computing, Graphic Era Hill University, Bell Road, Dehradun, Uttarakhand 248002, India
[5] Department of Computer Science and Engineering Graphic Era Deemed to be University, Dehradun 248002, India

* **Correspondence:** Email: c.swarup@seu.edu.sa, kumar.ankit@gla.ac.in, kamredudhamsingh@gmail.com.

Abstract: Rice is grown almost everywhere in the world, especially in Asian countries, because it is part of the diets of about half of the world's population. However, farmers and planting experts have faced several persistent agricultural obstacles for many years, including many rice diseases. Severe rice diseases might result in no grain harvest; hence, in the field of agriculture, a fast, automatic, less expensive, and reliable approach to identifying rice diseases is widely needed. This paper focuses on how to build a lightweight deep learning model to detect rice plant diseases more precisely. To achieve the above objective, we created our own CNN model "LW17" to detect rice plant disease more precisely in comparison to some of the pre-trained models, such as VGG19, InceptionV3, MobileNet, Xception, DenseNet201, etc. Using the proposed methodology, we took UCI datasets for disease detection and tested our model with different layers, different training–testing ratios, different pooling layers, different optimizers, different learning rates, and different epochs. The Light Weight 17 (LW17) model reduced the complexity and computation cost compared to other heavy deep learning models. We obtained the best accuracy of 93.75% with the LW17 model using max pooling with the "Adam" optimizer at a learning rate of 0.001. The model outperformed the other state-of-the-art models with a limited number of layers in the architecture.

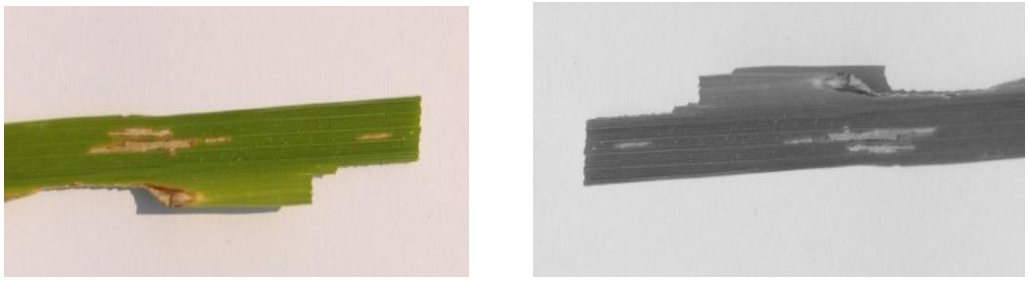
Keywords: rice diseases; transfer learning; UCI datasets; deep learning; InceptionV3; DenseNet201; VGG19

1. Introduction

More than half of the world population includes rice in their main meals, which makes it the most important food to cultivate and store. To increase the production of rice, it must be saved from various types of diseases [1]. Disease in rice is perhaps the biggest damaging factor that causes significant losses in rice production and the agricultural economy [2]. Additionally, diseases present significant dangers to food security. Obtaining data about the sustainable development and well-being of rice is featured in present-day agriculture. Various studies have indicated that rice plant diseases can be detected at an early stage by analyzing the leaf, stem, grain, etc., and during this stage, the crop can be saved by applying the necessary fertilizer. The traditional method used to identify rice plant diseases is based on human perception, which may differ from person to person and the skill of the person. This requires constant inspection by specialists, which may be restrictively costly on large farms [3]. Rice leaf-related infections frequently present dangers to the sustainable production of rice, influencing numerous farmers all over the planet. The early finding and suitable cure of rice leaf contamination is urgent for the sustainable development of rice plants to guarantee satisfactory inventory and food security for the quickly expanding population. Hence, machine-driven disease analysis frameworks could relieve the constraints of the customary strategies for leaf infection detection procedures that are tedious, error-prone, and costly. These days, machine learning algorithms have made rice leaf disease exceptionally famous [4].

Deep learning has become the most popular technique in the field of agriculture to predict diseases, test the soil, and make various other predictions, such as weather forecasting and fertilizer analysis. At the same time, these models have become very important to predict the two most harmful diseases of rice plants, i.e., fungal and bacterial diseases, which are the main causes for loss in the crop [5]. The traditional approaches to detect plant disease are by visual perception, by taking the advice of an expert, or by taking samples to the research lab for analysis, but all these approaches are time-consuming and not useful for early detection [6]. In addition, the prediction of disease by the naked eyes also needs mastery and perception, and the required time for analysis may vary from person to person. These issues may be resolved by an image-based AI approach to deal with the prediction and classification of plant diseases [7]. Particularly, in some countries, farmers might need to travel significant distances to contact agricultural specialists, which is tedious, costly, delays the forecast, and cannot be completed in a wide reach [8–10].

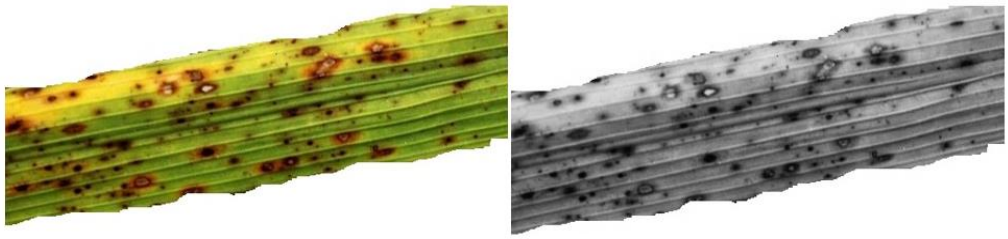
In this study, we detected three types of rice plant disease: bacterial leaf blight, brown spot, and leaf smut. Bacterial leaf blight, additionally called blight of rice, is a dangerous bacterial disease that is among the most horrendous diseases of developed rice. Leaves with yellow-white or bright yellow lines at the middle or edges may change or destroy their shape.



(a) (b)

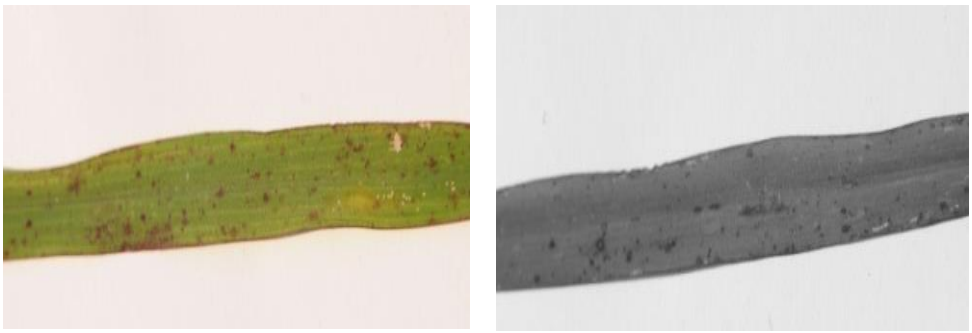
Figure 1. (a) RGB format image of bacterial leaf blight disease; (b) grayscale image of bacterial leaf blight disease.

Brown spot is a fungal disease that affects the color of the leaves. It is among the most dangerous of diseases and can damage all the leaves and the areas where the seed grows. Its symptoms are dim brown, unpredictable spots on the both upper and lower leaf surfaces, which will appear as yellow or light brown when held up to a backdrop illumination.



(a) (b)

Figure 2. (a) RGB format image of brown spot disease; (b) grayscale image of brown spot disease.



(a) (b)

Figure 3. (a) RGB format image of leaf smut disease; (b) grayscale image of leaf smut disease.

Leaf smut covers the entire leaf, but, to some degree, it is a minor infection of rice. Its

development creates marginally raised, precise, dark spots on both sides of the leaves. The reasons for leaf spots include organisms, microorganisms, and infections brought by environmental factors, such as natural circumstances, poison levels, and herbicide wounds.

In this work, we used deep transfer learning for the detection of rice plant diseases. The reuse of a pre-prepared model on a new issue is known as transfer learning in AI. A machine utilizes the information gained from an earlier task to increment prediction about another undertaking in transfer learning. For instance, it is possible to utilize the data acquired during preparation to recognize drinks while preparing a classifier to predict whether a picture contains wine. The information about a generally prepared ML model is moved to an alternate but firmly connected issue in transfer learning. There are various famous pre-prepared models (for example VGG16, Inception, ResNet50) available that are useful for defeating sampling lacks; they have proactively been trained on different images and can predict a variety of features. These models commonly have complex designs that are vital when interpreting the distinctions among hundreds or thousands of classes [11]. The complexity that offers a predictive limit with regards to the variety of items can be a hindrance for more simplistic tasks, as the pre-prepared model can overfit the information.

Segmentation is one of the most important pre-processing techniques in plant disease classification. The hue, saturation, and intensity (HSI) model and luminance, a, and b chrominance (Lab) models perform well regarding segmentation [12]. Some pretrained deep learning models have shown outstanding performance in the classification of rice plant disease from image datasets [13], whereas handmade CNN models may also create in some specific cases [14]. Transfer learning is another new concept used by many researchers to enhance the accuracy of classification with a lightweight model, such as Es-MbNet [15].

The contributions of this study are as follows:

1) We designed a 17-layer lightweight model to detect plant disease with low computational cost.
2) Various data augmentation techniques were applied to the benchmark dataset to increase the training performance of the model.
3) The performance of the designed model was compared with other existing models on the basis of various parameters.

2. Literature survey

Chen et al. [13] used a deep learning approach for solving the task of disease detection since it has shown outstanding performance in image processing and classification problems. They used a UCI Repository dataset. Combining the advantages of both, the DenseNet model pre-trained on ImageNet and the Inception modules were selected to be used in the network. This achieved an average predicting accuracy of no less than 94.07% on the public dataset. Even when multiple diseases were considered, the average accuracy reached 98.63% for the class prediction of rice disease images. Sony [16] used a convolutional neural network utilizing R language to find diseases in rice by utilizing pictures of infected leaves. The infection pictures gathered from the UCI Machine Learning Repository contained three types of diseases: bacterial leaf blight, brown spot, and leaf smut. By further developing the preparation pictures, they accomplished better outcomes with an accuracy of 86.6%. Al-Amin et al. [7] constructed a deep CNN model to predict four normal rice diseases, namely, bacterial leaf blight, brown spot, leaf smut, and blast. Using the testing dataset of more than 900 pictures of infections and solid leaves from the UCI Machine Learning Repository and following the method of 10-fold cross-

validation accomplished the most noteworthy accuracy of 97.40%. Their work presented a CNN-based model which provided 97.40% accurate results for different diseases of rice leaves. Wadhawan et al. [17] explored different techniques to recognize crop diseases in rice plants utilizing conventional image processing techniques and neural networks. They deployed an ML model that scanned the picture and, whether or not the leaf was tainted, estimated the area of contamination. The exactness of their identifications was around 85–86%. Ahmed et al. [18] presented a rice leaf disease discovery framework utilizing machine learning approaches. Three of the most widely recognized rice plant diseases, specifically, leaf smut, bacterial leaf blight, and brown spot, were identified in this work. Decision tree regression, after 10-fold cross-validation, accomplished a precision rate of more than 97% when applied to a test dataset that was collected from the UCI Machine Learning Repository [19]. Phadikar et al. [20] developed an automated system to classify two different types of rice diseases, including leaf brown spot and leaf blast diseases of rice plants, based on the morphological changes of the plants caused by the diseases. Radial distribution of the hue from the center to the boundary of the spot images was used as a feature to classify the diseases by Bayes and SVM Classifiers. Their identification scheme achieved an accuracy of 79.5%.

Azim et al. [21] proposed a model to detect three rice leaf diseases—bacterial leaf blight, brown spot, and leaf smut—in their paper. In their work, the backgrounds of the pictures were removed by the saturation threshold, while disease-affected areas were segmented using the hue threshold. The distinct components of color, shape, and surface space were separated from the impacted regions. Their model accomplished 86.58% accuracy on the rice leaf disease dataset from UCI [19]. Islam et al. [22] proposed a technique that applies threshold-based segmentation to separate the disease-affected areas on the rice leaves precisely. Three distinct CNN models—VGG16, ResNet50, and DenseNet121—have been used to group the diseases to decide the best model for such picture classification issues. DenseNet121 has been demonstrated as the best for this sort of image classification problem, with an achieved classification accuracy of 91.67%. Patidar et al. [23] proposed an accomplished model for the detection and classification of diseased rice leaves using a residual neural network. Three infections viz. leaf smut, bacterial leaf blight, and brown spot were distinguished by our framework, based on a dataset with 120 pictures (40 for every disease class). Their work accomplished an effective precision of 95.83%. The study by Teja et al. [24] was fundamentally centered around three different rice leaf diseases, i.e., brown spot, leaf smut, and bacterial leaf blight. The dataset for examination was acquired from the UCI, an ML library that contains pictures of three different leaf infections. Among various strategies, the best feasible arrangement was a shrewd blend of the advancement procedure taught to InceptionV3 and transfer learning, with an accuracy of 99.33%. Shrivastava et al. [25] utilized a pre-trained deep convolutional neural network (CNN) as a component extractor and support vector machine (SVM) as a classifier to classify rice plant disease. Their proposed model can characterize rice diseases with accuracy of 91.37% for an 80:20% training–testing ratio. Pushpa et al. [26] used the AlexNet model to develop a framework for detecting and identifying crop plant leaf diseases based on deep learning models. The result of the proposed approach was accomplished by distinguishing the boundaries, and the consequence of the CNN classifier-based features provided ideal accuracy in plant disease identification. Their proposed model accomplished an accuracy of 96.76%. Recently, deep learning models have been applied in various fields to detect targets in real-time images [27]. A combination of multi-scale feature selection [28] with different convolutional neural network models is playing an important role in various new studies [29–31]. These approaches are also very suitable for introducing automated machines and robots to different fields to save the time and effort of human

beings [32,33]. Some researchers have also used these techniques in other fields, such as for human face and handwriting recognition [34,35].

In the literature review of earlier works, we observed that for the detection of rice plant disease, many complex models have been used, which is a very time-consuming process. However, we did not find significant work that has been done with the grayscale dataset. So, our focus was on building a model that is less complex and has a lower computation cost while detecting diseases.

3. Methodology

Below is a pictorial representation of the workflow for the proposed methodology.

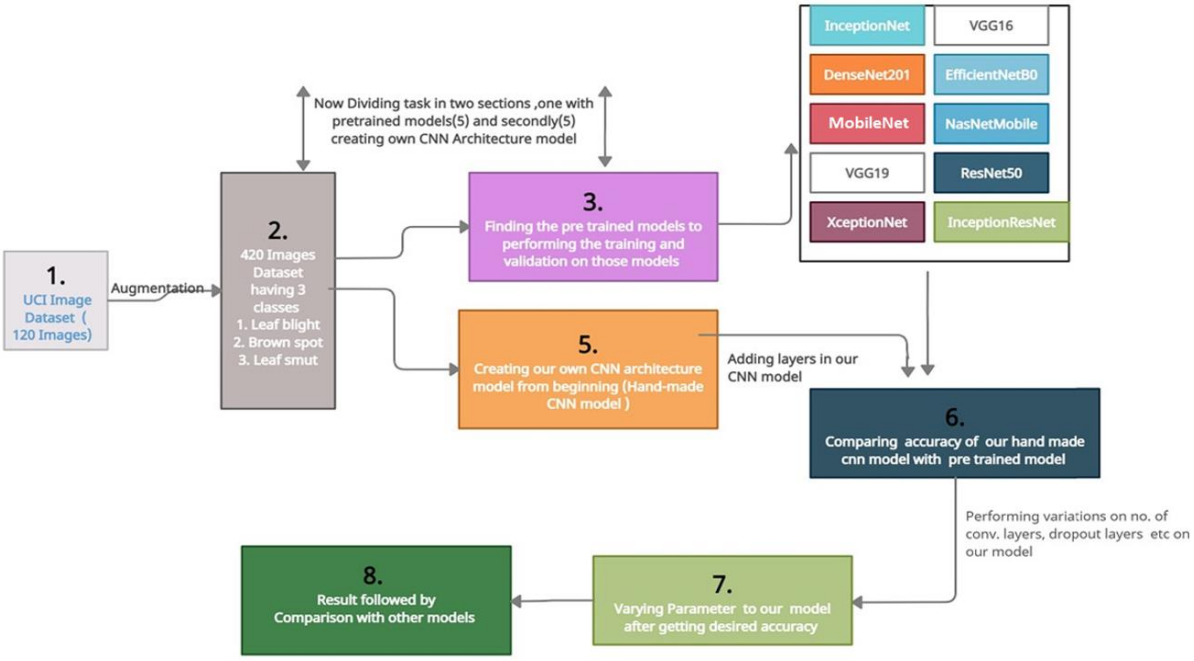


Figure 4. Pictorial representation of the workflow of our proposed methodology.

3.1. Dataset description

There are some limited public datasets available for plant disease detection. The most popular dataset of plant communities has more than 50,000 images of 38 classes, but it contains the leaves of various plants, and as our work was restricted to the diseases of rice plants only, we used another public dataset from the UCI Repository [13,24]. The UCI Repository is an international general database for the testing of machine learning algorithms (https://archive.ics.uci.edu/ml/datasets/rice+leaf+diseases). It comprises 120 images divided into three classes: bacterial leaf blight, brown spot, and leaf smut, with 40 images in each category.

3.2. Dataset preprocessing-data augmentation

Since this dataset is small for performing both training and validation, we performed data

augmentation to increase our dataset further to 420 images by performing rotation and zooming operations. All the images were resized to a fixed dimension of 120 × 120.

The rotation operation was performed by varying the angle of rotation from 15 degrees to 30 degrees by increasing the angle by 15 degrees after each iteration, and the zooming operation was performed by scaling the images by a factor of 1.5 in the x and y directions.

We manually performed all of the augmentations of the dataset, and as a result, we were able to gather 420 images of three different classes. Once we obtained the dataset, we then converted it into a NumPy array, which stores every pixel value of that specific image. Simultaneously, we added labelled the images as 0, 1, or 2.

3.3. Pre-trained model selection

To evaluate the performance of deep learning models [34], comparative experiments were performed. We tested a total of 10 pre-trained models with different training-testing ratios to select the best model. Thereafter, we varied different parameters, such as pooling, epochs, etc., to determine the best configuration for the selected model. Below, the models used in the methodology are described.

3.3.1. InceptionNet

Following the advancement in image processing by AlexNet, new ConvNets are made deeper and denser, such as the VGG model, the Dense model, etc. As these models have become deeper, their operations cost have also increased. Hence, to balance these costs, an InceptionNet was created with different versions—V1, V2, and V3. This InceptionNet was made by adding Inception modules. These Inception modules were made by using different combinations of the convolutional layer, pooling layer, dense layer, dropout, and fully connected layer. The performance of the InceptionNet on our dataset with variation in the number of epochs is shown in Table 2.

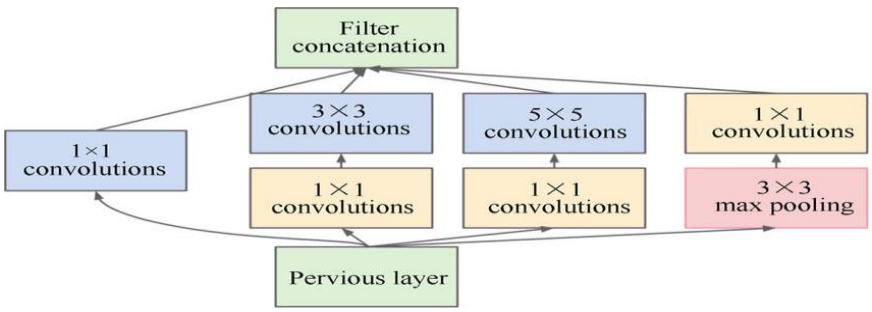


Figure 5. Block representation of a single Inception module from InceptionNet.

3.3.2. DenseNet201

As the name suggests, the DenseNet201 model is very densely made compared to other models. DenseNet201 is made up of dense blocks. The dense blocks in this study used ConvNets, and each layer in the dense block receives input from the other preceding layers, deepening to a total of 201 layers of these blocks. The performance of DenseNet201 on our dataset with variation in the number of epochs is shown in Table 2.

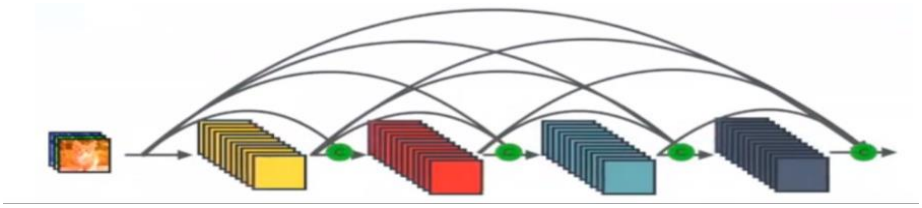


Figure 6. Block representation of a single dense module from DenseNet201.

3.3.3. MobileNet

MobileNet is basically used for the processing of mobile applications, and due to its light weight, it is becoming widely used by data scientists. It was the first image processing model created by TensorFlow. For convolution, MobileNet used two operations, Pointwise and Depthwise. The performance of MobileNet on our dataset with variation in the number of epochs is shown in Table 2.

3.3.4. VGG19

Visual Geometric Group 19 (VGG19) is a CNN-based model purpose-made for image classification. VGG19 is a 19-layer architecture. It comprises different blocks, including a convolutional layer, a pooling layer, and a dense layer, and each block is followed by a pooling layer that has a total count of 4. VGG19 performed exceptionally well in terms of accuracy and losses during the fitting of the model, as the yield accuracy increased within a few epochs compared to the other models. The performance of VGG19 on our dataset with variation in the number of epochs is shown in Table 2.

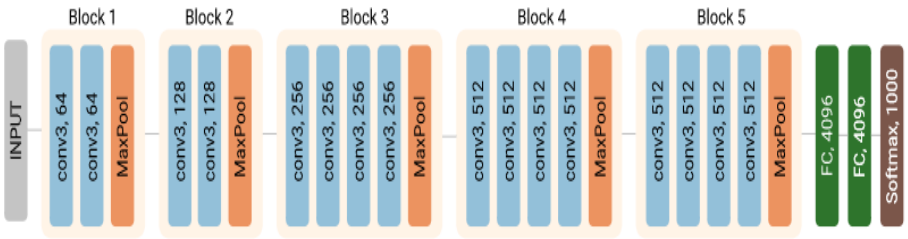


Figure 7. Block representation of a VGG19 model architecture with the ability to classify 1000 different classes.

3.3.5. Xception Net

XceptionNet was created by Google. The architecture of XceptionNet is 71 layers deep. Specifically, this model used the modified version of convolution in a depthwise format. While performing the depthwise operation of XceptionNet, no activation occurred in any of the intermediate layers. The performance of XceptionNet on our dataset with variation in the number of epochs is shown in Table 2.

3.3.6. VGG16

Visual Geometric Group 16 (VGG16) is a prior version of VGG19. In this study, VGG16 was used as a CNN-based model purposely created for image classification. VGG16 is a 16-layer architecture. It comprises different blocks of a convolutional layer, a pooling layer, and a dense layer, and each block is followed by a pooling layer that has a total count of 4. VGG16 performed exceptionally well in terms of accuracy and losses during the fitting of the model, as the yield accuracy increased within a few epochs as compared to the other models. For some datasets, VGG16 performed better than VGG19, and vice versa. The performance of VGG16 on our dataset with variation in the number of epochs is shown in Table 2.

3.3.7. EfficientNet

EfficientNet was not created by scientists; it was created with the help of artificial intelligence. EfficientNet is a convolutional neural network design and scaling method that uniformly scales all depth, breadth, and resolution dimensions using a compound coefficient. This EfficientNet randomly modified the network's breadth, depth, and resolution, and the EfficientNet scaling technique consistently scaled the network's breadth, depth, and resolution with a set of specified scaling coefficients. The performance of EfficientNet on our dataset with variation in the number of epochs is shown in Table 2.

3.3.8. NasNet Mobile

NasNet Mobile Neural architecture search was also developed by the Google Brain team. It was created for image processing and was trained with millions of images. It was developed to have the best CNN model for performing reinforcement learning. NasNet was used to find the best parameter for the CNN layers to attain the best accurate prediction results. The performance of NasNet on our dataset with variation in the number of epochs is shown in Table 2.

3.3.9. ResNet50

Residual Network 50 (ResNet50) is a type of neural network developed by researchers from Microsoft in the year 2015. This ResNet50 was a heavily dense model compared to InceptionNet and MobileNet, as this model contained five sub-blocks and each of these blocks was divided again into two other sub-blocks known as the identity block and convolution block. Both of these blocks had three convolution layers. The performance of ResNet50 on our dataset with variation in the number of epochs is shown in Table 2.

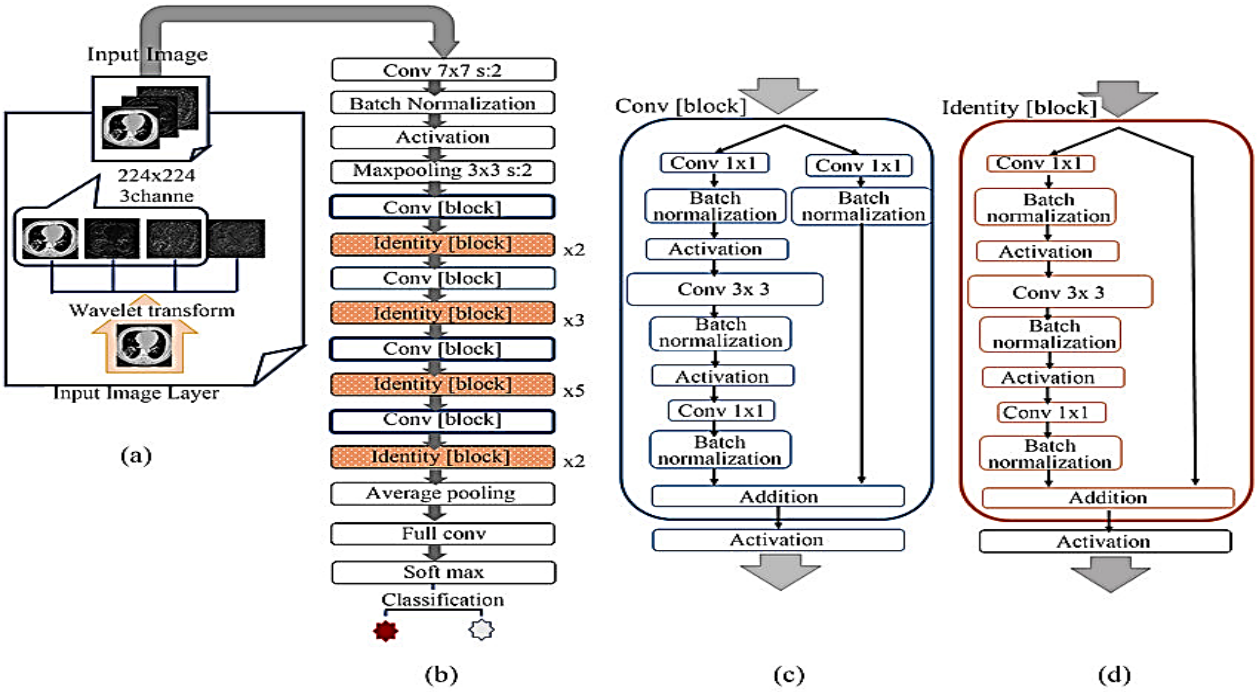


Figure 8. (a,b) ResNet50 architecture with dataset; (c) convolutional block; (d) identity block.

3.3.10. Inception ResNet

As the name suggests, Inception ResNet is the fusion of two neural networks—InceptionNet and Residual Network (ResNet). This neural network can predict thousands of different classes and has 164-layer architecture, which makes the model heavy and dense [24]. The performance of Inception ResNet on our dataset with variation of number of epochs is shown in Table 2.

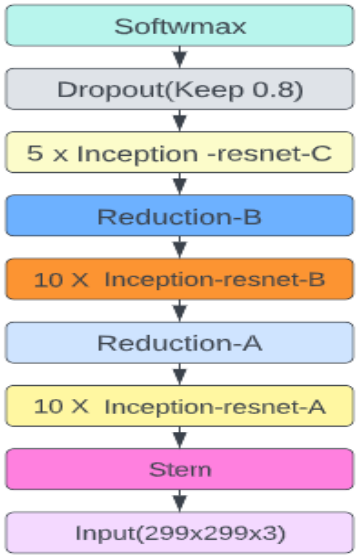


Figure 9. Block representation of how images are processed with InceptionResNetV1.

3.3.11. Model stacking

In some cases, to achieve transfer learning, we used the concept of model stacking. To connect the output of one deep learning model to the input of another, we used the output of the first model as the input for the second model. This was carried out by passing the output of the first model to the input layer of the second model. Before the model stacking, we ensured that the output of the first model had high-level feature representation so that it could be used as the input for the second model for further processing. The second model was then trained with this input to learn how to perform a specific task.

3.4. Light weight 17 (LW17) CNN model

3.4.1. Design of model

Input Layer: RGB is input to the model with a size of 120*120*3.

- **Block 1**: The first block starts with a convolutional layer (Conv1-1) through which a rice leaf image with a of size 118*118 pixels and 32 kernels, which have a size of 3*3, is passed. The activation used is "ReLu". After this, there is Conv2-2, which is also a convolutional layer with 32 kernels with a size of 3*3. The activation used here is also ReLu. In the last layer of this block, the image is pooled down using the max pooling layer, which has a size of 2*2.
- **Block 2**: The second block starts with Conv2-1, which takes input from the first block, with an input image size of 58*58 and an increased number of kernels with 64, which have a size of 3*3. The activation used is the ReLu function. Again, this layer is connected to Conv2-2, which also has the same parameters. Furthermore, it is connected to Conv2-3, which is also a convolutional layer. Using repetitive convolutional layers helped the model to achieve higher and higher accuracy. This Conv2-3 layer is connected to the second max pooling layer to pool down the size of the image, as the pooling size is 2*2. This means that the size of the image is decreased again by half, and at the end of the block, there is a dropout layer. The sole purpose of the dropout layer is to prevent overfitting of the model while training, with a dropout rate of 0.5.
- **Block 3**: The third block takes input from Block 2 with an input image size of 26*26. The first layer in this model is Conv3-1, a convolutional layer. The total number of kernels is 128, and the filter size is also the same, 3*3. The activation function is ReLu. This layer is further connected to Conv3-2 with the same parameters as that of Conv3-1. Conv3-2 is connected to the last convolutional layer of this layer, i.e., Conv3-3, with the same parameters as those of the other two layers. In the last layer, the image is pooled down by using a max pooling layer with a size of 2*2.
- **Block 4**: The fourth block is the last block where a convolutional layer is used. This layer takes input from the third block with a reduced image size of 10*10. The first layer is Conv4-1, and the total number of kernels is 256 with a size of 3*3. The activation function is ReLu, this layer is connected to the last convolutional layer of this model, Conv4-2, which has a total of 64 kernels with a size of 3*3, and the activation function as ReLu. Lastly, the image size is again pooled down to half by using a max pooling layer with a size of 2*2.
- **Block 5:** This block contains a single layer, which is a flattening layer. The flattening layer is

used to flatten the multidimensional array to a single dimensional array. Without this layer, it is not possible to classify objects using a 1D array.

- **Block 6:** This block contains the first dense layer of the model, named Dense-1, which takes input from the flattened layer. The purpose of adding a dense layer is because it feeds all the output coming from the last node to the current node and provides a single output. After this layer, we added a dropout layer with a dropout rate of 0.5, which reduces losses during training, potentially increasing the accuracy.
- **Block 7:** This block is similar to Block 6, as it also contains a dense layer and a dropout layer only. The sole purpose of this block to increase the accuracy of the model because the denser it is, the more noise-suppressed features can be extracted.
- **Block 8:** This is the last block of our proposed architecture. It has two layers, which are a dense layer and a softmax layer. This is the third and the last dense layer, with a total of three output nodes because we need to classify only three classes of rice plant disease. In the end, the softmax function is used to classify the images according to the classes. (The softmax function is used because the process is multiclass classification.)

To better explain the architecture, it is shown in Figure 10, and Table 1 represents the trainable parameters we used.

3.4.2. Parameters used

Loss Function: In this model, we used sparse categorical cross-entropy as the loss function since this loss function is used when more than one class of images is being classified. Loss function is useful because when the model is being fitted, it signifies how accurately the dataset is getting fitted while training the model. The higher the loss value, the more accurate the model's prediction.

$$CE = C * t * \log [f(s)] \tag{3.1}$$

here, CE is the cross-entropy, C is the the total number of output neurons, t is the target vector, and f(s) is the output value of the softmax function. Equation (3.1) shows the calculation of categorical cross-entropy.

Optimizer: The second key parameter while fitting the model is the optimizer. In this model, we used "Adam" as the optimizer, as it is faster and more efficient than some other optimizers, such as Adadelta, RMSprop, SGD, etc. The sole work of the optimizer is to modify the weights of each node after each epoch, and as a result, this will minimize the loss function. Adam uses a combination of two gradient descent algorithms, RMSprop and Momentum. This is how every block of the Light Weight 17 (LW17) model will perform its function whenever the image is passed through the input layer. Going further into the methodology, the training and validation of the RGB image datasets with the LW17 model are shown in Figure 10 below.

3.4.3. Model description

Table 1. Calculation of trainable parameters for LW17 CNN model.

Layer (type)	Output Shape	Param #
Conv2d 118 (Conv2D)	(None, 118, 118, 32)	896
Conv2d 119 (Conv2D)	(None, 116, 116, 32)	9248
Max pooling2d 56 (MaxPooling2D)	(None, 58, 58, 32)	0
Conv2d 120 (Conv2D)	(None, 56, 56, 64)	18,496
Conv2d 121 (Conv2D)	(None, 54, 54, 64)	36,928
Conv2d 122 (Conv2D)	(None, 52, 52, 64)	36,928
Max pooling2d 57 (MaxPooling2D)	(None, 26, 26, 64)	0
Dropout 28 (Dropout)	(None, 26, 26, 64)	0
Conv2d 123 (Conv2D)	(None, 24, 24, 128)	73,856
Conv2d 124 (Conv2D)	(None, 22, 22, 128)	147,584
Conv2d 125 (Conv2D)	(None, 20, 20, 128)	147,584
Max pooling2d 58 (MaxPooling2D)	(None, 10, 10, 128)	0
Conv2d 126 (Conv2D)	(None, 8, 8, 256)	295,168
Conv2d 127 (Conv2D)	(None, 6, 6, 256)	590,080
Max pooling2d 59 (MaxPooling2D)	(None, 3, 3, 256)	0
Flatten 14 (Flatten)	(None, 2304)	0
Dense 42 (Dense)	(None, 256)	590,080
Dropout 29 (Dropout)	(None, 256)	0
Dense 43 (Dense)	(None, 256)	65,792
Dropout 30 (Dropout)	(None, 256)	0
Dense 44 (Dense)	(None, 3)	771
Activation 14 (Activation)	(None, 3)	0

Notes: Total parameters: 2,013,411; trainable parameters: 2,013,411; non-trainable parameters: 0

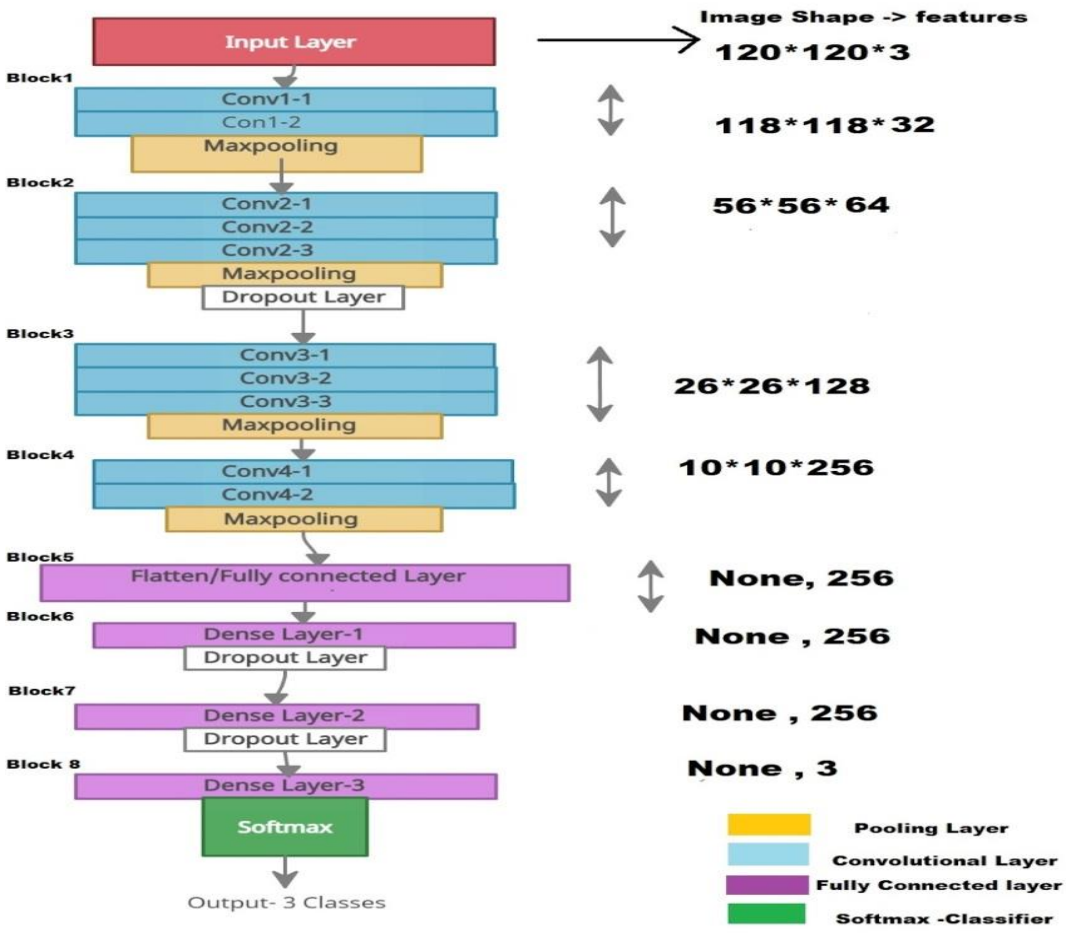


Figure 10. Architecture of LW17 CNN model.

After 17 layers, we also tried to increase the number of layers to 20 and 21, but we did not obtain any reasonable improvements in terms of accuracy. So, we stopped at 17 layers.

4. Result analysis

All of the experiments were performed on a Dell Precision tower 5810, Intel(R) CPU ES-2630, v4@2.20GHz, 64 GB, X64 Processor machine at the National Institute of Technology, Raipur. We used Python version 3.11 with a Jupyter Notebook, and the Scikit-learn, TensorFlow, Keras, Matplotlib, Pandas, etc. libraries were installed to perform the deep learning experiments.

We performed these experiments iteratively 10 times to obtain the average accuracy of each model. The accuracy of each model is shown in Tables 2 and 3.

Table 2. Accuracy of different deep learning models on color dataset for a random iteration.

Model	Number of Epochs	Training Accuracy	Validation Accuracy	Validation Loss
	5	90.06%	85.21%	1.1936
VGG-16	10	92.63%	87.43%	1.1302
	15	94.47%	89.78%	1.2301
	5	61.54%	51.28%	8.9556
INCEPTION 3	10	72.12%	58.97%	7.1420
	15	74.47%	59.10%	9.2561
	5	85.00%	50.18%	1.1704
MOBILENET	10	92.06%	52.21%	1.2332
	15	93.24%	52.63%	1.3489
	5	79.17%	79.1%	0.6808
DENSNET-201	10	89.74%	79.49%	0.6377
	15	90.47%	80.12%	0.5672
	5	66.53%	53.17%	3.6400
XCEPTION	10	69.50%	59.52%	4.3366
	15	70.47%	60.12%	4.5672
	5	91.60%	90.18%	0.2238
EFFICIENT NET	10	94.05%	94%	0.2682
	15	94.47%	94.10%	0.2561
	5	37.20%	33.33%	5.9619
INCEPTION RES	10	48.81%	45.24%	3.7276
	15	49.10%	45.71%	3.2682
	5	67.26%	55.95%	0.9370
NASNET MOBILE	10	70.24%	54.76%	1.2071
	15	71.36%	55.71%	1.1820
	5	93.15%	83.33%	0.3279
RESNET 50	10	94.35%	92.86%	0.3517
	15	**94.85%**	93.71%	0.3178
	5	92.26%	79.76%	0.8327
VGG-16	10	92.56%	84.52%	0.6207
	15	93.05%	85.54%	0.4178
	5	91.52%	87.50%	0.1711
LW17 CNN	10	90.81%	90.62%	0.1605
	15	**94.93%**	87.50%	0.2139

In Table 2, it can be observed that the ResNet-50 model obtained the highest accuracy, but the LW17 model had one-third of the layers of the ResNet-50 model and accuracy close to that model. Therefore, it can be a better option in terms of the complexity and cost of building a model.

Table 3. Comparison of average accuracy of LW17 model with different pertained models on color image dataset.

MODEL	Training Accuracy	Validation Accuracy	Validation Loss	Execution Time (in Sec)
Vgg19	90.50%	89.78%	1.2	218
InceptionV3	56.70%	50.30%	19	560
MobileNet	81.50%	77.80%	1.9	488
Densnet 201	77.70%	75.20%	0.72	540
XceptionNet	56.70%	52.20%	5.21	346
EfficientNet	89.31%	94.10%	0.27	246
InceptionResnet	49.10%	42.00%	46	389
NasnetMobile	70.24%	53.41%	1.22	458
Resnet50	94.85%	93.71%	0.37	548
Vgg16	92.44%	85.54%	0.68	110
LW17	**94.35%**	**93.75%**	**0.18**	**116**

Based on Table 3, we can say that our model performed better than any of the pre-trained models. Our model (LW17) gave the best validation accuracy in comparison to all the pre-trained models, i.e., 93.75%. when we compared the execution time. All other models (except VGG16) took more time than our model as they have more layers in their architectures.

When the LW17 model was tested with (approximately) 10% random testing data, the confusion matrix was obtained, as shown in Figure 11.

Class	1	2	3
1	12	0	0
2	2	15	1
3	0	0	14

Figure 11. Confusion matrix of the LW17 model on 10% random testing data.

Table 4. Comparison of the model with various parameters.

Class	Disease	Precision	Recall	F1-Score	Accuracy (%)
1	Bacterial Leaf Blight (BLB)	1.000	0.8571	0.923	100
2	Brown Spot	0.833	1.00	0.9088	83.33
3	Leaf Smut	1.00	0.933	0.965	100
-	Average	0.944	0.930	0.932	94.44

The precision, recall, F1-Score, (Eqs (4.1)–(4.3)), and execution time were also calculated to check the robustness and complexity of the proposed model. The results were calculated for each class and are shown in table 4. The average accuracy achieved on the testing dataset was 94.44%, which is much greater than the approach used by Chen et al. [13]. They achieved an average accuracy of 94.07% on the same public dataset, whereas another researcher, Patidar et al. [23], reported a highest accuracy of 95.83%, but they used the ResNet model for their experiments, which has more layers than our proposed model.

$$Precision = \frac{TP}{TP+FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP+FN} \tag{4.2}$$

$$F1 - Score = 2 * \frac{Precision*Recall}{Precision+Recall} \tag{4.3}$$

Table 5. Tabular comparison of our model with similar work.

Author	Accuracy (%)
A. Sony [16]	86.6
Radhika Wadhawan [17]	85
S. Phadikar et al. [20]	79.5
Azim, M. A. et al. [21]	86.58
Anam Islam et al. [22]	91.67
Vimal Shrivastava et al. [25]	91.37
Chen, J., et al., 2020 [36]	99.11
Chen, J. et al., 2021 [37]	98.50
Chen, J., et al., 2021 [38]	93.75
Chen, J., et al., 2021 [39]	99.14
Chen, J., et al., 2021 [40]	99.78
Chen, J., et al., 2021 [41]	99.67
Ruth J. A. et.al.,2022 [42]	99.00
Uma, R., & Meenakshi, A. ,2021 [43]	98.42
Gadekallu, T. R. et al., 2021 [44]	94.00
LW17 (Proposed Work)	**93.75**

Table 5 contains a comparison of the LW17 models with the other existing models. The comparison shows a significant improvement in the classification accuracy when using the LW17 model. In some of the other authors' work on the UCI dataset and the applied machine CNN model [16,17], Bayes' and SVM classifier [20,25], Boosted Decision Tree [21], convolution model [22], their training and testing accuracy was restricted to 91.67%, as the standard UCI dataset has a very small number of images so the system may not be trained enough with the same amount of data. Our system was trained with three times more images than the above-mentioned models, so its performance reached 93.75%. At the same time, when comparing the model with other standard techniques, such as transfer learning [36], the enhanced neural network model [38], MobileNetV2 [41], MobileNet with SE [40], attention-embedded lightweight network [37,39], and a recent meta-heuristic deep neural network (DNN) model, where features are optimized by a butterfly optimizer [42,43] and PCA [44], the performance of our system was restricted. Still, we succeeded in providing a model with fewer layers than these models and whose accuracy is close to that of these standard models.

5. Conclusions

Rice is an important food globally, and its production may be increased by reducing the effects of possible diseases. We observed that several computer-aided models exist, and every day, there is a new advancement in the models. In this study, we created our own CNN model from scratch to detect rice plant disease more precisely compared to pre-trained models, such as VGG19, InceptionV3, MobileNet, Xception, and DenseNet201. After trying different possible variations, we observed that our Model LW17 performed exceptionally well compared to other deep neural network models, with an accuracy of 93.75%. We obtained this result using max pooling, the optimizer "Adam", a learning rate of 0.001, and a training-to-validation ratio of 9: 1. We obtained the best accuracy for our model when we used the 17-layer architecture, which had almost a third of the layers compared to the ReNET50 model, whose highest accuracy was 93.85%. As the performance of the model was tested on a small dataset, in the future, we will test our model on other larger datasets to check its robustness. In addition, in the future, the performance of the proposed model may be compared with some more parameters.

Conflict of interest

All authors declare no conflict of interest with any other similar work.

References

1. Y. Lu, S. Yi, N. Zeng, Y. Liu, Y. Zhang, Identification of rice diseases using deep convolutional neural network, *Neurocomputing*, **267** (2017), 378–384. https://doi.org/10.1016/J.NEUCOM.2017.06.023
2. X. Wang, X. Zhang, G. Zhou, Automatic detection of rice disease using near infrared spectra technologies, *J. Indian Soc. Remote Sens.*, **45** (2017), 785–794. https://doi.org/10.1007/S12524-016-0638-6
3. V. K. Vishnoi, K. Kumar, B. Kumar, Plant disease detection using computational intelligence and image processing, *J. Plant Dis. Prot.*, ***128*** (2021), 19–53. https://doi.org/10.1007/s41348-020-00368-0
4. H. Al-Hiary, S. Bani-Ahmad, M. Reyalat, M. Braik, Z. AL-Rahamneh, Fast and accurate detection and classification of plant diseases, *Int. J. Comput. Appl.*, **17** (2011), 31–38. https://doi.org/10.5120/2183-2754

5. G. Kathiresan, M. Anirudh, M. Nagharjun, R. Karthik, Disease detection in rice leaves using transfer learning techniques, in *Journal of Physics: Conference Series IOP Publishing*, **1911** (2021), 012004. https://doi.org/10.1088/1742-6596/1911/1/012004
6. D. Al-Bashish, M. Braik, S. Bani-Ahmad, Detection and classification of leaf diseases using K-means-based segmentation and neural-networks-based classification, *Inf. Technol. J.*, **10** (2011), 267–275. https://doi.org/10.3923/ITJ.2011.267.275
7. M. Al-Amin, D. Z. Karim, T. A. Bushra, Prediction of rice disease from leaves using deep convolution neural network towards a digital agricultural system, in *22nd International Conference on Computer and Information Technology, ICCIT 2019*, (2019), 1–5. https://doi.org/10.1109/ICCIT48885.2019.9038229
8. M. E. Pothen, M. L. Pai, Detection of rice leaf diseases using image processing, in *Proceedings of the 4th International Conference on Computing Methodologies and Communication, ICCMC*, (2020), 424–430. https://doi.org/10.1109/ICCMC48092.2020.ICCMC-00080
9. A. Kaur, K. Guleria, N. K. Trivedi, Rice leaf disease detection: A review, in *6th International Conference on Signal Processing, Computing and Control (ISPCC)*, (2021), 418–422. https://doi.org/10.1109/ISPCC53510.2021.9609473
10. S. Ghosal, K. Sarkar, Rice leaf diseases classification using CNN with transfer learning, in *IEEE Calcutta Conference, CALCON 2020-Proceedings*, (2020), 230–236. https://doi.org/10.1109/CALCON49167.2020.9106423
11. *Plain English AI community*, Available from https://ai.plainenglish.io/building-and-training-a-convolutional-neural-network-cnn-from-scratch-9a64bcc62c1.
12. Y. A. Nanehkaran, D. Zhang, J. Chen, Y. Tian, N. Al-Nabhan, Recognition of plant leaf diseases based on computer vision, *J. Ambient Intell. Human. Comput.*, (2020), 1–18. https://doi.org/10.1007/s12652-020-02505-x
13. J. Chen, D. Zhang, Y. A. Nanehkaran, D. Li, Detection of rice plant diseases based on deep transfer learning, *J. Sci. Food Agric.*, **100** (2020), 3246–3256. https://doi.org/10.1002/jsfa.10365
14. J. Chen, J. Chen, D. Zhang, Y. A. Nanehkaran, Y. Sun, A cognitive vision method for the detection of plant disease images, *Mach. Vis. Appl.*, **32** (2021), 1–18. https://doi.org/10.1007/s00138-020-01150-w
15. J. Chen, A. Zeb, Y.A. Nanehkaran, D. Zhang, Stacking ensemble model of deep learning for plant disease recognition, *J. Ambient Intell. Human. Comput.*, (2022), 1–14. https://doi.org/10.1007/s12652-022-04334-6
16. A. Sony, Prediction of rice diseases using convolutional neural network (in Rstudio), *Int. J. Innovat. Sci. Res. Technol.*, **4** (2019), 595–602. https://ijisrt.com/assets/upload/files/IJISRT19DEC446.pdf.
17. R. Wadhawan, M. Garg, A. K. Sahani, Rice plant leaf disease detection and severity estimation, in *IEEE 15th International Conference on Industrial and Information Systems, ICIIS 2020 - Proceedings*, (2020), 455–459. https://doi.org/10.1109/ICIIS51140.2020.9342653
18. K. Ahmed, T. R. Shahidi, S. M. I. Alam, S. Momen, Rice leaf disease detection using machine learning techniques, in *International Conference on Sustainable Technologies for Industry 4.0*, *STI*, (2019). https://doi.org/10.1109/STI47673.2019.9068096
19. *UCI Machine Learning Repository*, Rice leaf diseases data set, Available from: https://archive.ics.uci.edu/ml/datasets/Rice+Leaf+Diseases. Accessed: 2019-09-27.
20. S. Phadikar, Classification of rice leaf diseases based on morphological changes, *Int. J. Inf. Electron. Eng.*, **2** (2012), 460–463. https://doi.org/10.7763/IJIEE.2012.V2.137

21. M. A. Azim, M. K. Islam, M. M. Rahman, F. Jahan, An effective feature extraction method for rice leaf disease classification, *Telkomnika (Telecommunication Computing Electronics and Control)*, **19** (2021), 463–470. https://doi.org/10.12928/TELKOMNIKA.V19I2.16488
22. A. Islam, R. Islam, S. M. R. Haque, S. M. M. Islam, M. Ashik, I. Khan, Rice leaf disease recognition using local threshold-based segmentation and deep CNN, *Intell. Syst. Appl.*, **5** (2021), 35–45. https://doi.org/10.5815/ijisa.2021.05.04
23. S. Patidar, A. Pandey, B. A. Shirish, A. Sriram, Rice Plant Disease Detection and Classification Using Deep Residual Learning, in *Machine Learning, Image Processing, Network Security and Data Sciences: Second International Conference, MIND 2020, Silchar, India, July 30-31*, (2020), 278–293. https://doi.org/10.1007/978-981-15-6315-7_23
24. K. U. A. R. Teja, B. P. V. Reddy, L. R. Kesara, K. D. P. Kowshik, L. A. Panchaparvala, Transfer Learning based Rice Leaf Disease Classification with Inception-V3, in *International Conference on Smart Generation Computing, Communication and Networking, SMART GENCON*, (2021). https://doi.org/10.1109/SMARTGENCON51891.2021.9645888
25. V. K. Shrivastava, M. K. Pradhan, S. Minz, M. P. Thakur, Rice plant disease classification using transfer learning of deep convolutional neural networks, *Int. Arch. Photogramm., Remote Sens. Spat. Inf. Sci*, **42** (2019), 631–635. https://doi.org/10.5194/isprs-archives-XLII-3-W6-631-2019
26. B. R. Pushpa, A. Ashok, A. V. S. Hari, Plant disease detection and classification using deep learning model, in *Proceedings of the 3rd International Conference on Inventive Research in Computing Applications*, ICIRCA, (2021), 1285–1291. https://doi.org/10.1109/ICIRCA51532.2021.9544729
27. L. Huang, Q. Fu, M. He, D. Jiang, Z. Hao, Detection algorithm of safety helmet wearing based on deep learning, *Concurrency Comput.: Pract. Exper.*, **33** (2021), e6234, https://doi.org/10.1002/cpe.6234
28. J. Yun, D. Jiang, Y. Liu, Y. Sun, B. Tao, J. Kong, et al. Real-time target detection method based on lightweight convolutional neural network, *Front. Bioeng. Biotechnol.*, **10** (2022), 861286, https://doi.org/10.3389/fbioe.2022. 861286
29. L. Huang, C. Chen, J. Yun, Y. Sun, J. Tian, Z. Hao, et al. Multi-scale feature fusion convolutional neural network for indoor small target detection, *Front. Neurorobit.*, **16** (2022), 881021. https://doi.org/10.3389/fnbot.2022.881021
30. L. Huang, Z. Xiang, J. Yun, Y. Sun, Y. Liu, D. Jiang, H. Ma, et al. Target detection based on two-stream convolution neural network with self-powered sensors information, *IEEE Sens. J.*, 2022, https://doi.org/10.1109/JSEN.2022. 3220341
31. Y. Liu, D. Jiang, C. Xu, Y. Sun, G. Jiang, B. Tao, et al, Deep learning based 3D target detection for indoor scenes, *Appl. Intell.*, (2022), 1–14. https://doi.org/10.1007/s10489-022-03888-4
32. X. Zhang, J. Liu, J. Feng, Y. Liu, Z. Ju, Effective capture of non-graspable objects for space robots using geometric cage pairs, *IEEE/ASME Trans. Mechatron.*, **25** (2020), 95–107. https://doi.org/10.1109/TMECH.2019.2952552
33. Q. Gao, J. Liu, Z. Ju, X. Zhang, Dual-hand detection for human-robot interaction by a parallel network based on hand detection and body pose estimation, *IEEE Trans. Ind. Electron.*, **66** (2019), 9663–9672. https://doi.org/10.1109/TIE.2019.2898624
34. Y. A. Nanehkaran, D. Zhang, S. Salimi, J. Chen, Y. Tian, N. Al-Nabhan, Analysis and comparison of machine learning classifiers and deep neural network techniques for recognition of Farsi handwritten digits, *J. Supercomput.*, **77** (2021), 3193–3222. https://doi.org/10.1007/s11227-020-03388-7

35. Y. A. Nanehkaran, J. Chen, S. Salimi, D. Zhang, a pragmatic convolutional bagging ensemble learning for recognition of Farsi handwritten digits, *J. Supercomput.*, **7** (2021), 13474–13493. https://doi.org/10.1007/s11227-021-03822-4
36. J. Chen, D. Zhang, Y. A. Nanehkaran, identifying plant diseases using deep transfer learning and enhanced lightweight network, *Multimedia Tools Appl.*, **7** (2020), 31497–31515. https://doi.org/10.1007/s11042-020-09669-w
37. J. Chen, W. Wang, D. Zhang, A. Zeb, Y. A. Nanehkaran, Attention-embedded lightweight network for maize disease recognition, *Plant Pathol.*, **7** (2021), 630–642. https://doi.org/10.1111/ppa.13322
38. J. Chen, J. Chen, D. Zhang, Y. A. Nanehkaran, Y. Sun, A cognitive vision method for the detection of plant disease images, *Mach. Vis. Appl.*, **32** (2021), 1–18. https://doi.org/10.1007/s00138-020-01150-w
39. J. Chen, W. Chen, A. Zeb, D. Zhang, Y. A. Nanehkaran, Crop pest recognition using attention-embedded lightweight network under field conditions, *Appl. Entomol. Zool.*, **56** (2021), 427–442. https://doi.org/10.1007/s13355-021-00732-y
40. J. Chen, D. Zhang, M. Suzauddola, Y. A. Nanehkaran, Y. Sun, Identification of plant disease images via a squeeze-and-excitation MobileNet model and twice transfer learning, *IET Image Process.*, **15** (2021), 1115–1127. https://doi.org/10.1049/ipr2.12090
41. J. Chen, D. Zhang, A. Zeb, Y. A. Nanehkaran, Identification of rice plant diseases using lightweight attention networks, *Expert Syst. Appl.*, **169** (2021), 114514. https://doi.org/10.1016/j.eswa.2020.114514
42. J. A. Ruth, R. Uma, A. Meenakshi, P. Ramkumar, Meta-heuristic based deep learning model for leaf diseases detection, *Neural Process. Lett.*, **54** (2022), 5693–5709. https://doi.org/10.1007/s11063-022-10880-z
43. R. Uma, A. Meenakshi, Apple leaf disease identification based on optimized deep neural network, in *Handbook of Research on Deep Learning-Based Image Analysis Under Constrained and Unconstrained Environments*, (2021), 167–185. https://doi.org/10.4018/978-1-7998-6690-9
44. T. R. Gadekallu, D. S. Rajput, M. P. K. Reddy, K. Lakshmanna, S. Bhattacharya, S. Singh, et al., A novel PCA–whale optimization-based deep neural network model for classification of tomato plant diseases using GPU, *J. Real-Time Image Process.*, **18** (2021), 1383–1396. https://doi.org/10.1007/s11554-020-00987-8

AIMS Press