Smart Agricultural Technology 4 (2023) 100195

Contents lists available at ScienceDirect

Smart Agricultural Technology

journal homepage: www.journals.elsevier.com/smart-agricultural-technology

Application of smartphone-image processing and transfer learning for rice disease and nutrient deficiency detection

Anshuman Nayak, Somsubhra Chakraborty *, Dillip Kumar Swain

Agricultural and Food Engineering Department, IIT Kharagpur, India, 721302

ARTICLE INFO

Editor by: Dr Spyros Fountas

Keywords:
Image processing
Smartphone
Application
Rice disease detection
Rice nutrient deficiency detection
Transfer learning

ABSTRACT

The earliest detection of plant disease is the primary concern of the farming community. The availability of advanced digital cameras and smartphones with improved image acquisition modes and deep learning methods like convolutional neural networks (CNN) can detect plant disease with high accuracy. This study used 2259 smartphone images of various rice (*Oryza sativa*) plant parts under various classes and 250 real-time validation images for classifying 12 rice diseases and nutrient deficiency symptoms. Different image segmentation techniques like foreground extraction were used to segment affected portions. Additionally, optimization of models and procedures to use them in smartphones with offline working capabilities have been described. Furthermore, a dynamic framework has been developed and demonstrated where the server trains on unseen image data to improve the classification performance and updates the model on falling below a certain threshold level. A comparison was made across different models used for image classification with many supporting metrics to select the best model for transfer learning. The top four performing models were DenseNet201, Xception, MobileNetV2, and ResNet50, with validation accuracies of 0.9803, 0.9778, 0.9756, and 0.9718, respectively. The ResNet50 model was found to best among all for cloud architectures, while MobileNetV2 appeared as the best model for the smartphone application. Finally, the android application, "Rice Disease Detector" compiled with the MobileNetV2 model was tested for multiple disease occurrences in a single capture. More research is warranted to test the application for smartphones with variable configurations.

1. Introduction

Around 9% of crop and livestock production is lost annually due to disease and pests worldwide [1]. Farmers lose more than 6 billion USD every year in India due to crop disease and pest attacks. Plant disease is responsible for approximately 15–25% loss of agricultural products in countries like India [2]. Plant disease and deficiency detection require on-field diagnosis in a stipulated time to save the crop. In the current era of digital agriculture, conventional plant disease detection by visual inspection and laboratory-based analysis methods appear time-consuming and laborious. Sometimes disease detection based on visual inspection includes bias, misconceptions, and errors [3]; making it difficult for inexperienced and young farmers. Consequently, the detection of plant diseases and nutrient deficiencies needs expert staffing. It is also difficult to detect the nutrient deficiency symptoms and differentiate them from diseases due to shared symptoms, especially in standing water conditions in transplanted rice fields. Therefore, there is a pressing demand for an accurate and rapid diagnosis of plant diseases to lessen quantitative and qualitative losses.

Mahlein et al. [4] concluded that imaging systems perform better than non-imaging systems for plant disease detection. So far, much of the research on plant disease detection and crop health monitoring has been done on unmanned aerial vehicle (UAV) image processing and satellite imaging techniques [5]. UAVs usually cover a large area in a short time; however, the chances of error are generally more due to low-resolution and noise-prone images [6]. Using UAVs for the aerial survey also needs permission from governing bodies in certain countries and locations [7]. The cost incurred in the entire process becomes prohibitively high [8]. These methods are also climate-dependent and need expert supervision for complete decision-making. Hence, the procurement of UAVs and high-resolution imagery for small farm holders in developing countries has become economically impractical [9]. Hence, a climate and location-independent, low-cost, affordable, and high-resolution imaging device can solve these problems.

Over the last decade, the development of machine learning, deep learning, and image processing techniques has shown potential for the

* Corresponding author at: Agricultural and Food Engineering Department, Indian Institute of Technology Kharagpur, India.
E-mail address: somsubhra@agfe.iitkgp.ac.in (S. Chakraborty).

https://doi.org/10.1016/j.atech.2023.100195
Received 12 December 2022; Received in revised form 4 February 2023; Accepted 4 February 2023
Available online 6 February 2023

development of rapid and real-time tools for the analysis of images taken using high-resolution, multi-featured cameras [10,11]. Recently, Johannes et al. [12] used a mobile capture device and statistical inference method for multiple plant disease detection. Picon et al. [13] used a deep residual neural network-based algorithm to classify three European endemic wheat diseases. Lu et al. [14] proposed a pattern recognition method and CNN to identify rice diseases. Hasan et al. [15] and Sethy et al. [16] used deep feature-based transfer learning and a support vector machine classifier to classify rice diseases. Furthermore, Cruz et al. [17] used deep learning-based models for end-to-end detection of yellow symptoms in grapevines. Among different deep learning methods, the use of convolutional neural networks (CNN) has shown remarkable performance for plant disease detection [18]. Even though various supervised and unsupervised learning methods have shown promise for rapid and automated plant disease detection, they are fraught with constraints like limited data handling, complicated feature extraction following appropriate image segmentation, etc. [19]. All previous studies have shown that image segmentation, feature extraction, transfer learning, and image classification techniques can automate plant disease detection, decreasing the time and cost incurred.

In developing countries like India, the availability and affordability of the internet and agro-advisory systems is a common issue in remote areas and among farming communities. Consequently, using smartphones for taking high-resolution images seems to be an affordable and promising alternative [20], as these devices can go close to the affected plants and take pictures at high resolutions. This also mitigates legal issues and expert supervision. The increased availability of average and low-ranged android smartphones among the rural populations and field-level advisors (costing ~ 105–180 USD) put forward a way of using optimized image processing and deep learning algorithms in smartphones without needing a dedicated server. The results can be obtained instantly, and there is almost no need for any communication system to convey them. However, while smartphone-clicked images usually have higher resolutions, processing takes more time, especially through algorithms like Grab-Cut segmentation [21]. Therefore, the objectives of this study were: 1) to evaluate the combination of image processing and transfer learning for smartphone image-based detection of rice diseases and nutrient deficiency symptoms and 2) to develop an android application to implement the real-time disease and nutrient deficiency symptoms detection.

The following research gaps have been addressed in this article:

- Demonstration of the usability of low-cost grab-cut foreground segmentation and other standard image processing techniques for effective CNN training, considering the presence of unusual noise, varying aging, and illumination conditions, which were not addressed by previous studies,
- Evaluation of the performance of 11 popular CNN models (DenseNet 201, EfficientNetB0, InceptionV3, MobileNet, MobileNetV2, NASNetMobile, ResNet 101, ResNet 50, VGG16, VGG19, Xception) using progressively freezing convolution and stochastic depth optimization technique for 12 rice disease and deficiency classes with one healthy class,
- Consideration of other affected plant parts like panicle, neck, and apex with leaf for effective disease and nutrient deficiency detection,
- Development of an optimized and offline android application with a monitoring server for low-range smartphones to implement real-time detection, and
- Calibration of the best model for the detection of multiple diseases and deficiencies occurrences in a single capture, which is a common field condition and remains unexplored by previous studies.

2. Materials and methods

2.1. Description of rice diseases detected

Plant diseases are of two types biotic and abiotic. Diseases reveal some similar sets of characteristics in various plant parts. Some characteristics are local to certain parts of the field, and others may spread over an entire group of areas. This characteristic analysis plays a significant role in plant disease detection and future diagnosis. Characteristics of plant disease and deficiency symptoms can be described as follows.

2.1.1. Rice blast

The symptoms of rice blast include lesions that can be found on all parts of the plant, including leaves, leaf collars, necks, panicles, pedicels, and seeds [22]. However, the most common and diagnostic symptom, diamond-shaped lesions, of rice blast occur on the leaves [23]. Here we have considered three kinds of rice blast disease: apex blast, leaf blast, and neck blast.

2.1.2. Bacterial blight

Rice bacterial blight is caused by the bacterium Xanthomonas oryzae PV. Oryzae. The initial signs of rice with bacterial leaf blight are water-soaked lesions at the edges and toward the tip of the leaf blades. Lesions grow larger and release milky sap that dries and turns a yellowish color. This is followed by characteristic, grayish-white lesions on the leaves. This last stage of infection precedes the drying out and death of the foliage within 2–3 weeks.

2.1.3. Brown spot

Brown spot disease of rice is caused by the fungus *Helminthosporium Oryzae*. The period of infection can be from the seedling to the milking stage. The disease first appears as minute brown dots and later becomes cylindrical or oval to circular on leaves. The yield loss can be 50% of the total attainable production in severe cases.

2.1.4. Stem rot

Rice stem rot is a fungal disease caused by the pathogen *Sclerotium oryzae*. This disease affects water-sown rice plants and usually becomes noticeable in the early tillering stage. Symptoms begin as small, rectangular black lesions on the leaf sheaths at the waterline of flooded rice fields.

2.1.5. Leaf burn

Rice leaf burn is a fungal disease caused by *Alternaria Padwickii*. Symptoms appear on leaves and ripening grains. Circular to oval spots with dark brown margins appear on the leaves. These spots show many light brown to white spots at the center of the leaves.

2.1.6. Leaf smut

Leaf smut is caused by the fungus *Entyloma Oryzae*. The fungus creates angular black spots on both sides of the leaves. Heavily infected turns yellow and leaf tips turn to die and turn gray. This appears in the late growing season and causes loss.

2.1.7. Rice tungro virus

This is caused by the Rice tungro bacilliform virus. Plants affected by tungro show stunting and reduced tillering. Leaves become yellow or orange-yellow; they may also develop rust-colored spots. Discoloration can be seen from leaf tip to blade.

2.2. Detected deficiency symptoms of rice

2.2.1. Nitrogen deficiency

Nitrogen deficiency in rice plants causes reduced plant growth, yellowing of leaf tissues, or sometimes yellowing of the entire plant. This

mainly occurs in Nitrogen demanding growth stages such as tillering, panicle initiation, etc.

2.2.2. Phosphorus deficiency

Phosphorus deficiency in rice can be characterized by stunted, narrow, short, very erect, and dark green to bluish leaves. Symptoms can be seen in older leaves, even if young leaves seem to be healthy.

2.2.3. Potassium deficiency

Potassium deficiency has almost no effect on rice tillering. Observed symptoms usually can be seen in older tissue; these cause leaf tips to turn yellow-brown, eventually drying up and becoming necrotic.

2.3. Image acquisition

A total of 2259 images were used, consisting of 950 images from the plant village data set [24] and 1309 images collected from multiple fields from different villages near to Indian Institute of Technology, Kharagpur, India, and various locations of Baragarh and Puri districts of Odisha, India. Images were taken under different daylight conditions and different stages of rice, starting from early tillering to panicle initiation. Plant village images (1881 × 1881 pixel resolution) were captured using a 12-megapixel (MP) iPhone X camera (f/1.8). A Xiaomi Redmi Note 8 camera with a 48-MP camera (f/1.8, 26 mm (wide), ½", 0.8 μm and a phase-detection autofocus 2 MP camera, f/2.4) with a macro mode setting was used for collecting the rest of the leaf images. Some images were also taken by placing a hand as a background for the rice leaf, where the macro mode feature was unavailable. The original data set includes various images for 12 different disease classes, with a minimum number of 118 images for rice leaf smut and a maximum number of 223 images for the brown spot. The numbers of original images under each class are listed in Table 1. Images of the healthy leaf are also taken from fields with diseased leaf images. Fig. 1a shows the field-collected images for various disease classes.

2.4. Image pre-processing

Images collected from the field were processed via resizing, foreground segmentation, and dataset balancing. Images collected were scaled down to 300 × 300 pixels since many pre-built TensorFlow neural network models feature a maximum dimensional requirement of 299 × 299 pixels (Xception Model). Image resizing was executed by resampling using the pixel-area relationship as it offers moire' effect-free results [25]. Foreground segmentation of images was achieved using the Grab-Cut algorithm, which employs a Gaussian Mixture Model (GMM) [26]. A region of interest (ROI) was selected containing the desired foreground, while all the pixels outside the ROI were labeled as background. Next, GMM was used to create a pixel distribution based on the hard-labeled data. Subsequently, a pixel-distribution graph was created, taking pixels as a node. Additional two nodes called source (to connect foreground pixels) and sink (to connect background pixels) were added. Each node was connected either to the source or sink, depending upon its probability of falling into the foreground or background. Then a min-cut algorithm was used to separate the source and sink with a minimum cost function. Fig. 1b shows the Grab Cut segmented and resized images under various classes.

Table 1
Number of images under each class.

Class name	Number of images under each class
Apex blast	190
Bacterial blight	190
Brown spot	223
Gudi rotten	220
Healthy	150
Leaf blast	147
Leaf burn	190
Leaf smut	118
Neck blast paddy	190
Nitrogen deficiency	207
Phosphorus deficiency	159
Potassium deficiency	145
Tungro	130

Since the image dataset featured an unequal distribution of images for each class, this could lead to poor classification performance. Consequently, random oversampling of data for all classes was executed, resulting in 600 images under each class. Random oversampling includes processes like random rotation, random noise injection, and flipping of images to generate a standard data set for training and validating neural networks. Test data sets for each class were taken from fields and processed with resizing and foreground extraction, to test the on-field effectiveness of the model. Fig. 1c illustrates the images generated after random augmentation of image classes.

2.5. Feature extraction and neural network training

Choosing the best model for classifying images is the most crucial step for creating real-time and offline compatible mobile applications. Many models include multi-step layers for feature detection and training. In the case of pre-available models, top layers were excluded to avoid the loss of information they might contain during previous training. Some of the convolution layers were also frozen progressively before compiling the model [27]. Freezing the convolution layers accelerates the training process. Convolutional layers were also used with the stochastic depth concept to randomly drop a subset of layers for each mini-batch and replace them with identity functions. This operation helped to improve the performance time by reducing the size of the neural network during training, although a full network was used during validation [28]. A description of the various models used in this project is given in Table 2. The entire data set was divided randomly into 70% training and 30% validation sets. Images were resized to the desired dimension of the model. Image data were cached and buffered for faster access during training and validation. Notably, caching image data requires enough primary memory (RAM) for the smooth training of neural network models on cloud platforms; otherwise, it will lead to an unexpected termination of the process. For faster convergence of images, the datasets were normalized. Adam's optimizer was used to optimize the model since it costs less for a benchmark of 50 iterations over the entire dataset than all other available alternatives [29]. Fig. 2 demonstrates the steps involved in training neural network models.

2.6. Smartphone interface

An optimized smartphone user interface can reduce many costs associated with the process. Hence, the "Rice Disease Detector" application interface was designed with a specific area-based wireframe on the screen to reduce this time and increase user-friendliness. The application was provided with two interfaces: one to select an image from the gallery and the other to take an image using the camera. The application camera compresses the image to 80% of the original quality. Fig. 3a demonstrates the user interface of the android application.

2.7. Image processing and neural network simulation on a smartphone

The smartphone interface was designed using React Native framework [30]. The size of the application depends upon the modules used, which were optimized using Emscripten, an LLVM/Clang-based compiler, and only the required functionalities were compiled into JavaScript. Tensorflow JS was used for simulating the model. The model trained in the cloud was compiled, saved, and converted to binaries and weights. Tensorflow JS created a compiled model from these binaries, which was used for image classification. Fig. 3b demonstrates the flow of the simulation process on a smartphone interface.

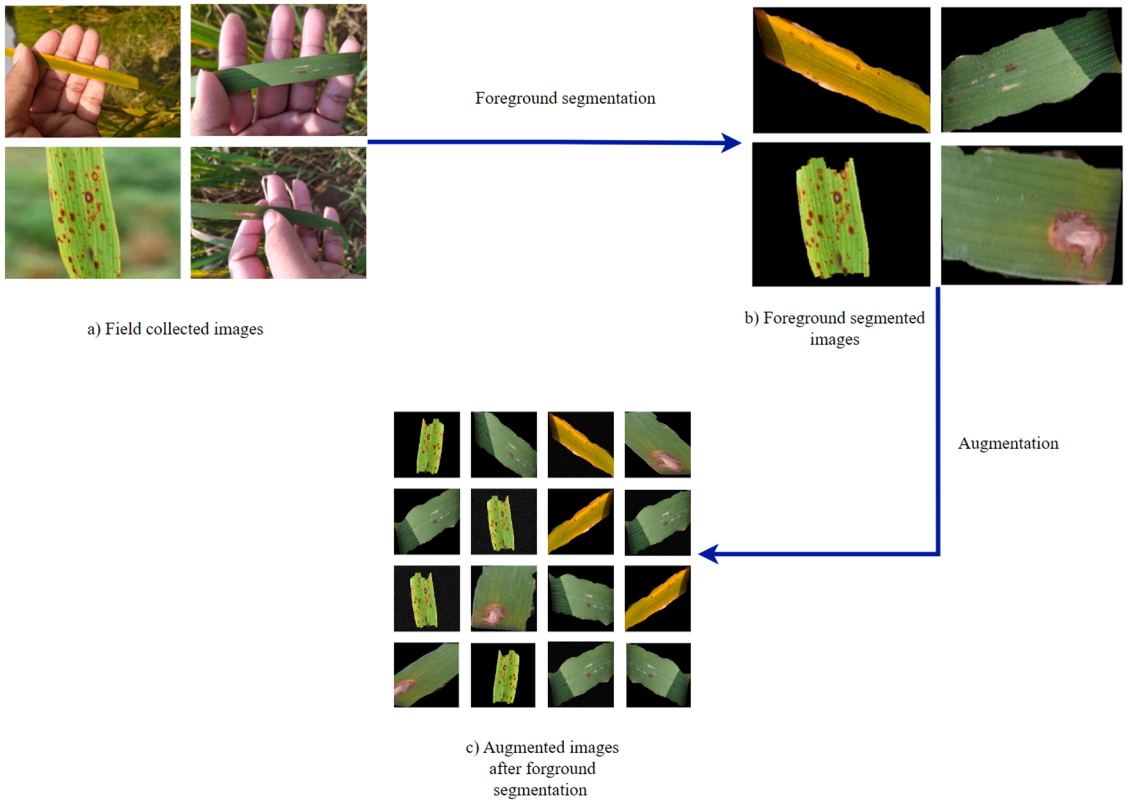


Fig. 1. Flowchart showing image pre-processing before model training: a) field-collected images for various diseases, b) foreground segmented and resized images, and c) augmented images after foreground segmentation.

Table 2
Technical description of deep learning models used for rice disease classification.

Model name	Top 1 accuracy	Top 5 accuracy	Depth	Compiled size in (MB[a])	Input dimension (pixel)	Time for step over GPU[b] (ms)
Densenet 201	77.30%	93.60%	402	80	224×224	6.7
EfficientNetB0	77.10%	93.30%	132	29	224×224	4.9
InceptionV3	77.90%	93.70%	189	92	299×299	6.9
MobileNet	70.40%	89.50%	55	16	224×224	3.4
MobileNetV2	71.30%	90.10%	105	14	224×224	3.8
NASNetMobile	74.40%	91.90%	389	23	224×224	6.7
ResNet 101	76.40%	92.80%	209	171	224×224	5.2
ResNet 50	74.90%	92.10%	107	98	224×224	4.6
VGG16	71.30%	90.10%	16	528	224×224	4.2
VGG19	71.30%	90.00%	19	549	224×224	4.4
Xception	79.00%	94.50%	81	88	299×299	8.1

[a] MB Megabyte.
[b] GPU Graphics processing unit.

2.8. Software and hardware specifications

The image processing has been done using OpenCV (version 4.5.2.54) and the scikit-image (version 0.18.1) python library. The source code for model simulation, training, and validation has been written using Tensorflow 2.0 python library. Results have been visualized using the Matplotlib library (version 3.4.2). Simulation of models has been done in a cloud instance with Nvidia P100 graphics processing unit (GPU), having specifications of 16 GB memory, 1.32 GHz clock speed, and 9.3 TFLOPS computational speed. The central processing unit (CPU) was Dual-core Intel Xeon with 13 GB RAM. This configuration is freely available with Kaggle's standard usage cloud instance. Mobile application testing has been performed using Xiaomi Redmi Note 8 android smartphone with Mali-G52 MC2 GPU and 8 cores CPU.

2.9. Model validation and testing

Model validation was done along with training in the cloud platform. The model was tested both on the smartphone device and the cloud with 250 field-collected images. Field-collected images were acquired considering the variability in illumination and daylight condition. In general, many agricultural fields exhibit more than one disease on plant parts. Accordingly, a set of images were developed to mimic such conditions by merging random numbers (two to four) of pre-collected images. A total of 100 images consisting of more than one disease or deficiency symptom were generated to test the classification model.

2.10. Performance metrics and dynamic model improvements

The developed Android application installed on a smartphone device can perform all the operations in both offline and online modes.

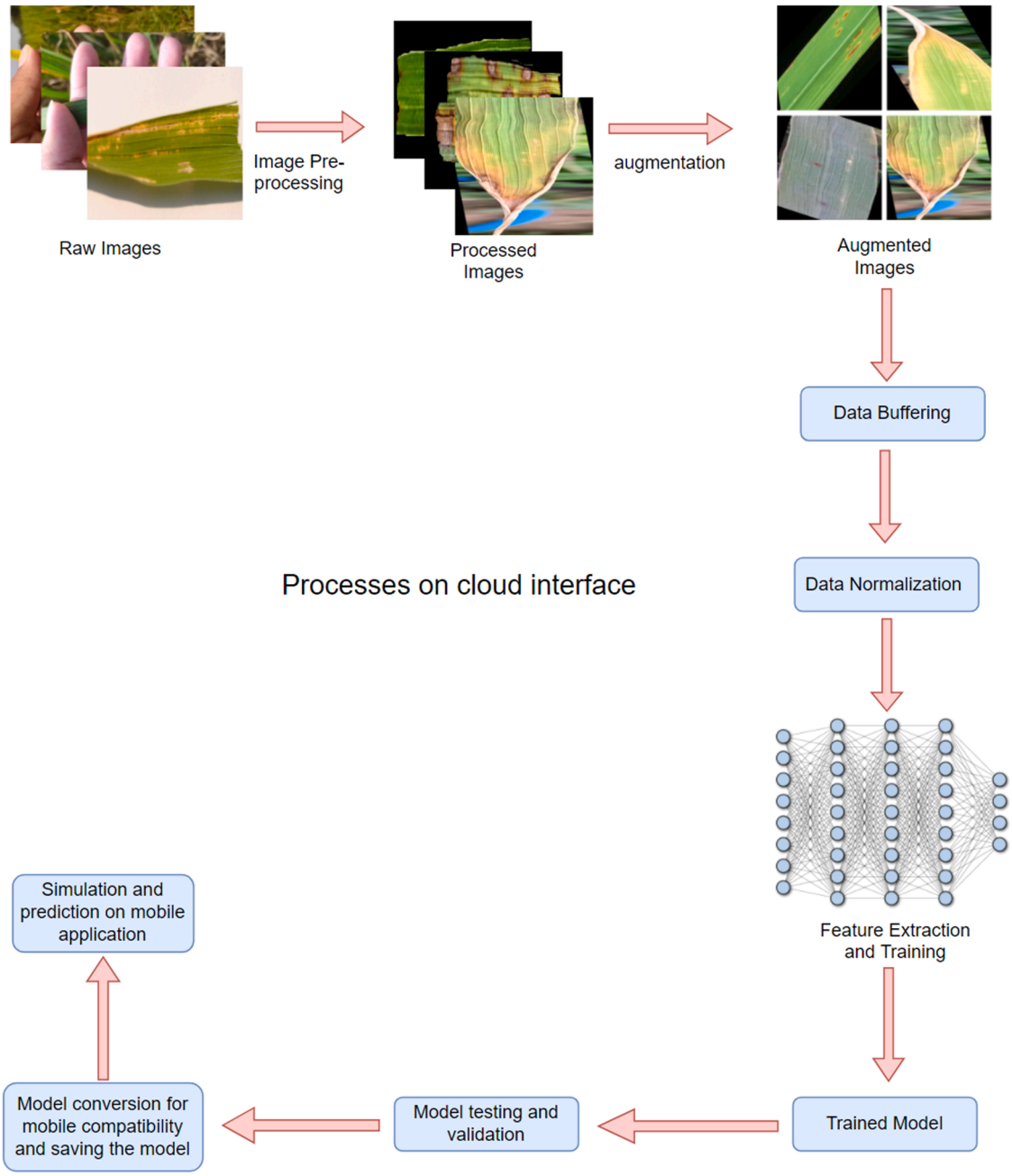


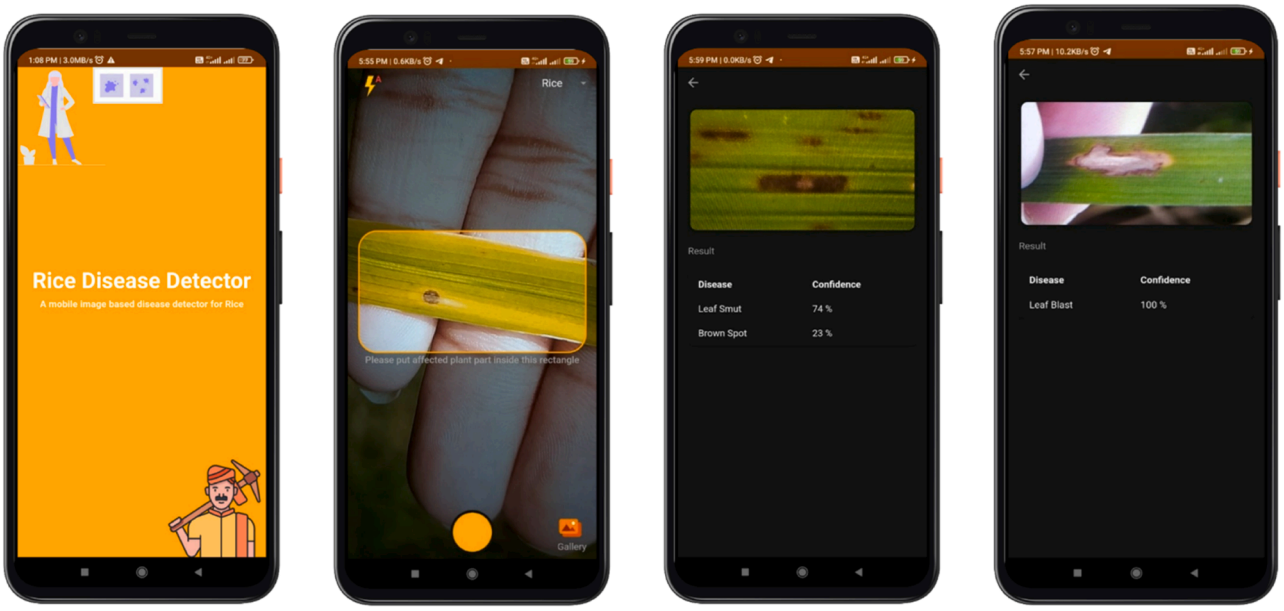
Fig. 2. Flowchart showing the processes of feature extraction and training of the neural network.

Application logs were maintained and saved locally to the device. Whenever the internet was available, logs were uploaded to the server, performance thresholds were set, and whenever prediction went below a certain level, it was uploaded to the server. The model was trained for 50 iterations over the entire data set, and its performance was observed with nine scalar metrics. Since the accuracy of any neural network model cannot be judged by only one conclusive metric, metrics like F-beta (F1 and F2), average metrics, Matthew's correlation coefficient (MCC), and false-positive rate (FPR) were used. Moreover, receiver operating characteristics (ROC) which show how the model can distinguish between classes, were examined. The ROC- area under the curve (AUC) curve was plotted using the one versus rest approach as it is a multiclass classification problem. The accuracy of the topmost and top 5 layers was tested with 600 images available for training and validation. Finally, the model was selected for comparison by considering the step times, time to predict in smartphone devices, compiled size, accuracy over top layers, and other metrics. A description of the metrics is presented in Table 3.

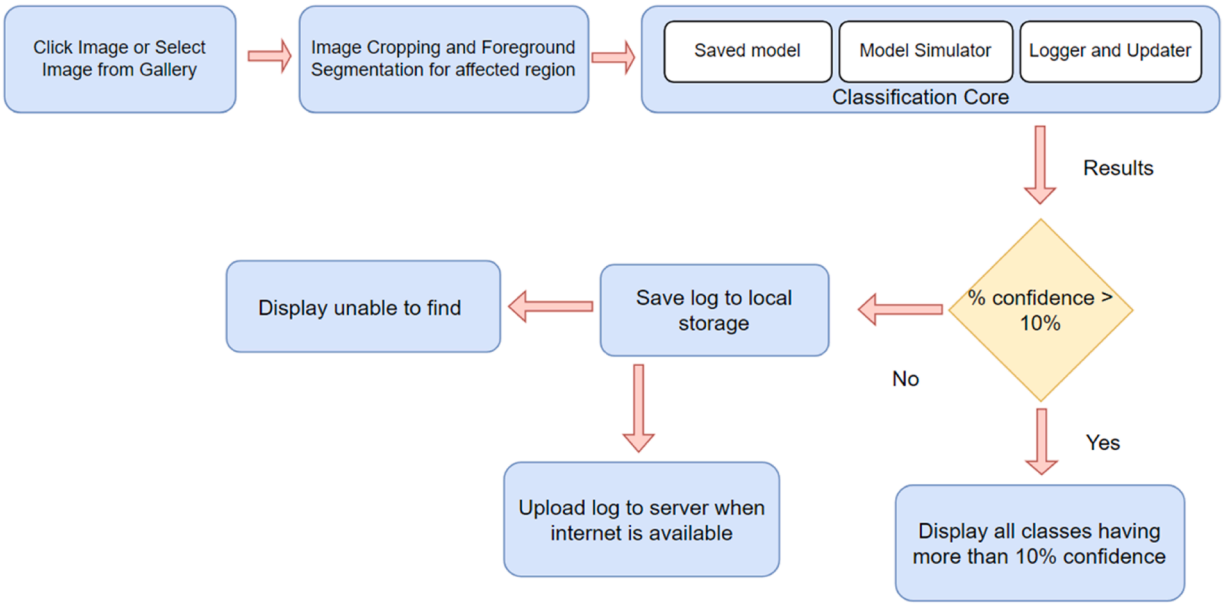
For multiple disease conditions, class confidence below a previously observed threshold value (10%) was rejected, and confidence above the value was supplied to the user [31]. The server automatically updated the model (over the air update system) without manually editing the application whenever an update was available. Fig. 3a demonstrates a condition where the application has correctly predicted multiple diseases present in the image.

3. Results and discussions

This study evaluated the performance of 11 popular CNN models, out of which four were small (Efficient B0, MobileNet, MobileNetV2, and NASNetMobile), and seven were moderate to large (DenseNet 201, InceptionV3, ResNet 101, ResNet 50, VGG16, VGG19, and Xception) in



(a)



(b)

Fig. 3. a) The user interface of the smartphone Android application, with a demonstration of detected diseases. b) Flowchart showing the working of the "Rice Disease Detector" Android application.

size after compilation. Along with other performance metrics (Table 3), model size, time to train, and time to predict in smartphone devices were also considered for selecting the best model. Training and validation metrics showing considerable variations are presented in Tables 4 and 5, respectively. For experimental purposes, model training and validation were performed over 50 epochs with 600 processed images under each class, with training to validation ratio of 70:30. In addition, the "Rice Disease Detector" application's performance was tested on over 100 real-field images with one or more disease classes in the ROI.

3.1. Cloud simulation results based on accuracy and loss metrics

In this study, CNN models EfficientB0, Inception V3, and VGG19 did not exhibit a constant value of loss or accuracy over 50 epochs. Although they represented the lower values of loss (~ 0.02) and higher values of accuracy (> 0.90) at any epoch, the patterns of getting these values were inconsistent. Contrariwise, DenseNet 201, MobileNet, MobileNetV2, NasNetMobile, RestNet101, ResNet50, and Xception models exhibited training accuracy of 1 and loss close to 0 (ranging from 0.00001514 to 0.00009434) after training with a fixed number of epochs (Table 4). The validation accuracy of all these models appeared in the range of 0.9658 to 0.9803, while DenseNet 201 model yielded the highest accuracy. The

Table 3
Metrics used for performance measurement of neural network.

Metrics	Formula
Specificity (Precision)	$\frac{TP^a}{TP + FP^b}$
Sensitivity (TPR[e] / Recall)	$\frac{TP}{TP + FN^c}$
False positive rate (FPR[f])	$\frac{FP}{FP + TN^d}$
Average metrics	$\frac{Specificity + Sensitivity}{2}$
F-Beta score	$(1 + \beta^{2,g}) \times \frac{Precision \times Recall}{(\beta^2 \times Precision) + Recall}$
Matthew's correlation coefficient (MCC)	$\frac{TN \times TP - FN \times FP}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$
Accuracy	$\frac{TN + TP}{TP + FP + TN + FN}$

[a]TP True positive; [b]FP False positive; [c]FN False negative; [d]TN True negative.
[e] TPR True positive rate.
[f] FPR False positive rate, [g]β = Coefficient of weightage given to precision than that of recall.

number of epochs after which a constant and steady performance was recorded was considered the optimal number of epochs required for training. The number of optimal epochs varied from 20 for the ResNet 50 model to 42 for DenseNet 201.

The VGG16 model exhibited an unusual behavior of constant lower accuracy and loss in training after 26 epochs. The model accuracy suddenly declined from 0.8560 at the 25th epoch to 0.2804 at the 26th epoch and continued in that range. The loss metrics for this model abruptly increased from 0.4713 at the 25th epoch to 238.7969 at the 26th epoch. The loss declined after the 26th epoch and continued in a lower range. This unusual behavior is illustrated in Fig 4.

3.2. Cloud simulation results based on F-beta, FPR, and MCC values

In this study, two F-beta metrics, F1 and F2 were used. Higher and consistent values of training and validation F1 score and lower values of F2 score were observed after cloud simulation with optimal epochs for many models (DenseNet 201, MobileNet, MobileNetV2, NasNetMobile, RestNet101, ResNet50, and Xception) (Tables 4 and 5). The value of the F1 score was 0.9585 and (0.9704 - 0.9901) for training and validation sets respectively for these seven models.

The FPR score for training and validation did not show considerable deviation across the consistent models trained after their respective optimal epochs. However, there was a smaller increase in FPR values in the validation set than in training sets from (0.0769 to 0.0820). Notably, MCC is the most critical measure of any classification model, as it measures the performance of the classifier, taking into consideration True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values. In this experiment, the values of MCC for consistent models were > 0.79 and 0.67 for training and validation sets, respectively. The unusual behavior of the VGG16 model continued over this set of performance metrics, which can be seen in Fig. 4.

Table 4
Training set metrics for various deep learning model used to classify plant diseases and deficiency symptoms.

Model name	Optimal epochs to train	Training loss after optimal epochs	Training accuracy	Training average metrics	Training MCC[a]	Training FPR[b]	Training F1 score	Training F2 Score	Time to train in cloud* (Seconds)
DenseNet201	42	0.00006183	1	0.9615	0.8320	0.0769	0.9585	0.0012	2700
EfficientB0	Inconsistent	–	–	–	–	–	–	–	2100.3
InceptionV3	Inconsistent	–	–	–	–	–	–	–	2196.05
MobileNet	36	0.00001487	1	0.9615	0.8024	0.0769	0.9585	0.0012	1286
MobileNetV2	36	0.00002688	1	0.9615	0.8024	0.0769	0.9585	0.0012	1402.8
NasNetMobile	28	0.00001514	1	0.9615	0.8437	0.0769	0.9585	0.0012	1302.6
ResNet101	23	0.00006329	1	0.9615	0.7992	0.0769	0.9885	0.0012	2596.85
ResNet50	20	0.00009434	1	0.9615	0.8001	0.0769	0.9585	0.0012	1586.5
VGG16	Inconsistent	–	–	–	–	–	–	–	2012.35
VGG19	Inconsistent	–	–	–	–	–	–	–	2305
Xception	23	0.00003975	1	0.9615	0.8102	0.0769	0.9584	0.0012	6320.95

[a] MCC Matthew's correlation coefficient.
[b] FPR False positive rate; [c]NPV Negative predictive value.
* Specification of the cloud: GPU- Nvidia P100, Memory 16 GB, 1.32 GHz, 9.3 TFLOPS; CPU: - Dual-core Intel Xeon with 13 GB RAM.

Table 5
Validation set metrics for various deep learning model used to classify plant diseases and deficiency symptoms.

Model Name	Validation loss	Validation accuracy	Validation average metrics	Validation MCC[a]	Validation FPR[b]	Validation F1 Score	Validation F2 Score	AUC[d] Score	Time to predict in a mobile device* (Seconds)
DenseNet201	0.0997	0.9803	0.9694	0.6745	0.0820	0.9733	0.0012	0.9909	7.04
EfficientB0	Inconsistent	–	–	–	–	–	–	–	–
InceptionV3	Inconsistent	–	–	–	–	–	–	–	–
MobileNet	0.1909	0.9658	0.9763	0.8004	0.0820	0.9791	0.0012	0.9845	3.57
MobileNetV2	0.1656	0.9756	0.9677	0.8001	0.0820	0.9831	0.0012	0.9868	3.09
NasNetMobile	0.1733	0.9731	0.9695	0.8237	0.0820	0.9884	0.0012	0.9851	3.23
ResNet101	0.1852	0.9705	0.9745	0.7521	0.0820	0.9901	0.0012	0.9820	4.56
ResNet50	0.1810	0.9718	0.9731	0.8001	0.0820	0.9885	0.0012	0.9846	4.21
VGG16	2.7037	0.0781	–	–	–	–	–	–	–
VGG19	Inconsistent	–	–	–	–	–	–	–	–
Xception	0.1351	0.9778	0.9647	0.8100	0.0820	0.9704	0.0012	0.9880	9.03

[a] MCC Matthew's correlation coefficient.
[b] FPR False positive rate [c] NPV Negative predictive value.
[d] AUC Area under curve.
* Mobile device Specifications: - Xiaomi Redmi Note 8 with Mali-G52 MC2 GPU and 8 cores CPU (2 × 2.0 GHz Cortex-A75 & 6 × 1.8 GHz Cortex-A55).

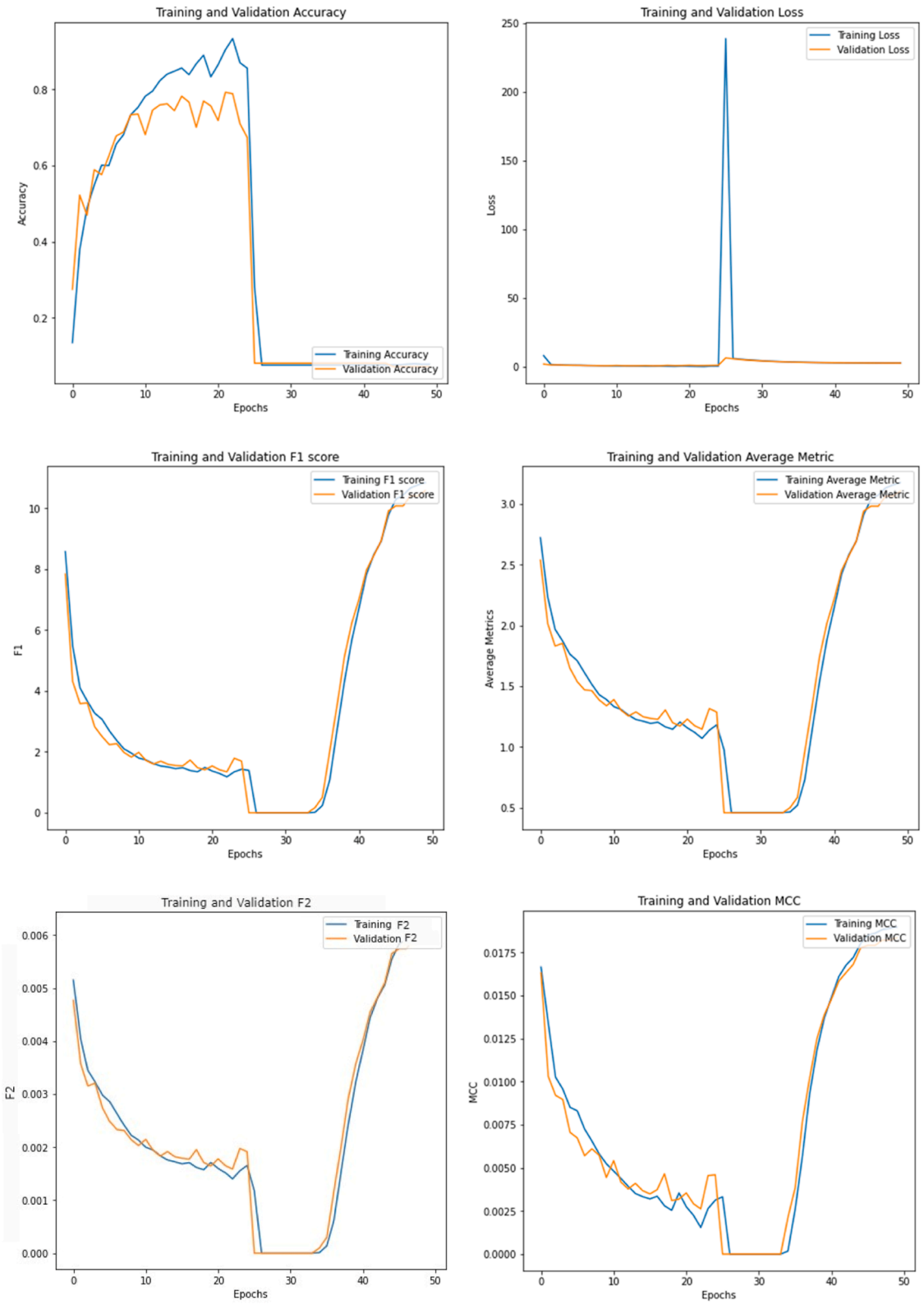


Fig. 4. Performance metrics of the VGG16 model.

3.3. Cloud and smartphone simulation results based on the time taken and auc scores

The Xception model was the slowest training model and took at least three times (6320.95 s) more time on average than other models (Table 5). Conversely, MobileNet and MobileNet V2 were the fastest training models, with 1286 and 1403 s of training duration, respectively. The mobile simulation time was recorded by taking an average of time taken for getting the prediction of 10 images. Again, the trend was similar, with a minimum time of 3.09 s taken by MobileNetV2 and a maximum time of 9.03 s for the Xception model.

The AUC (Area under the curve) score is an essential factor, along

with other metrics for choosing the best model. AUC score defines the ability of a classification algorithm to differentiate among its classes. One versus rest approach has been followed for AUC metrics calculation. The DenseNet 201 model exhibited the highest AUC score of 0.9909, and ResNet 50 model exhibited the lowest AUC score of 0.9820 (Table 5). There was less variation in the AUC score among the consistent models.

3.4. Practical implications

The accuracy and loss metrics observed for DenseNet 201, MobileNet, MobileNetV2, NasNetMobile, RestNet101, ResNet50, and Xception models across the training and validation set revealed that the models were not overfitted. Also, validation performance worsened than training, which was inevitable because of unseen noises and image features in the former. Observed higher F1 and lower values of F2 illustrated that the consistent models had a good balance between precision and recall. Indeed, the values of F1 were found to be better than that observed by Sethy et al. [16]. The observed lower values of FPR close to 0 showed the correct predictive ability of the trained model over the validation set. The values of FPR were ten times higher than those reported by Sethy et al. [16] (ranging from 0.004 to 0.01). This could be due to the increased number of images, classes under consideration, and noises induced to simulate the field condition. The MCC values were found to be close to the multilayer perceptron classifier explained by Patil and Burkpalli [32] (0.70–0.85). Observed values of MCC in both training and validation sets were close to 1, implying that the consistent models were good classifiers.

The time models take to be trained and predict the result largely depends upon the convolution operation performed. Since the images were resized, the background was eliminated using the grab-cut algorithm, and approaches like freezing the convolutional layer with stochastic depth optimization were performed in CNN, models performed faster than other conventional methods [13,16,33,34]. However, the total time taken by Xception to train over 50 epochs differed from that of Sethy et al. [16] (9750 s) and Wang et al. [35] (2400 s), which can be attributed to the difference in smartphone hardware configurations, the number of images trained, and image optimization techniques.

The performance of DenseNet 201 and Xception was found to be better than that of other researchers [16,36] and the best among all other models under observation. Nevertheless, since these two models were the slowest performer in the cloud and mobile interfaces, they were not considered for application development. Contrariwise, EfficientB0, Inception V3, and VGG19 models showed inconsistent performance, indicating the need for training over more than 50 epochs or removal of some other fully connected layers to get a steady performance. A similar model performance was recorded by Sethy et al. [16] for VGG16. The performance of model VGG16 worsened after 25 epochs, perhaps due to model overfitting, controlled training, and random depth optimization performed by a stochastic optimization technique. Indeed, VGG16 exhibited an unusual graphical representation representing the model's overfitting (Fig. 4). The trend observed by VGG 16 model supported the observation by Min et al. [37]. These results highlighted the ineffectiveness of the VGG16 model with the proposed methodology.

ResNet50 outperformed all models considering the performance and number of epochs to be trained to get a consistent performance. Among all models, MobileNetV2 performed faster on low-end smartphone devices with a competing validation accuracy of 97.56% and an AUC score of 0.98. Classification performances of these models are given in Table 6. The confusion matrix and ROC of the ResNet 50 and MobileNetV2 models are shown in Figs. 5 and 6, respectively. The ROC curve of ResNet 50 and MobileNetV2 for different classes corroborated the findings of Picon et al. [13], indicating a better classification of plant disease classes. Overall, ResNet50 and MobileNet V2 appeared as the best models considering cloud simulation and other factors.

Table 6
Classification metrics for two best models.

Class Names	Precision	Recall	F1-Score	Support
MobileNetV2				
Apex blast	0.99	1.00	1.00	172
Bacterial blight	0.95	0.99	0.97	166
Brown spot	0.98	0.92	0.95	186
Gudi rotten	1.00	1.00	1.00	207
Healthy	0.91	0.99	0.95	174
Leaf blast	0.95	0.90	0.93	186
Leaf burn	0.99	1.00	0.99	190
Leaf smut	0.99	0.99	0.99	174
Neck blast paddy	1.00	0.98	0.99	181
Nitrogen deficiency	0.97	0.99	0.98	164
Phosphorus deficiency	0.96	0.96	0.96	168
Potassium deficiency	0.96	0.96	0.96	184
Tungro	1.00	1.00	1.00	188
ResNet50				
Apex blast	0.99	1.00	1.00	172
Bacterial blight	0.95	0.97	0.96	166
Brown spot	0.95	0.91	0.93	186
Gudi rotten	1.00	1.00	1.00	207
Healthy	0.87	0.98	0.92	174
Leaf blast	0.96	0.89	0.92	186
Leaf burn	1.00	1.00	1.00	190
Leaf smut	1.00	1.00	1.00	174
Neck blast paddy	1.00	0.98	0.99	181
Nitrogen deficiency	0.98	0.98	0.98	164
Phosphorus deficiency	0.98	0.95	0.96	168
Potassium deficiency	0.95	0.98	0.97	184
Tungro	1.00	1.00	1.00	188

3.5. Choosing the model suitable for smartphone devices and cloud applications

The major difficulties in designing an offline-capable smartphone application are application size, limited and variable smartphone hardware resources, varying camera resolution and modes associated with the camera, bugs, error tracking, and handling unforeseen data [38]. Hence, the compiled size of ResNet50 is large enough (107 MB) for a smartphone device to handle. For cloud-based applications, ResNet 50 can be used as a dynamic model that improves itself with unforeseen image data with the least cost of computation, given it requires lower optimal epochs. ResNet 50 model performed the best among all tested models, corroborating the results of other researchers [13,16]. Nevertheless, the MobileNet V2 model was found to be the model with readily acceptable training and validation performance metrics and a size of 14 Mb suitable for working on smartphone devices. All the performance metrics of MobileNetV2 are illustrated in Fig. 7.

3.6. Testing application for multiple disease conditions

The "Rice Disease Detector" application's usability in a condition of multiple disease occurrences was tested using over 100 images (34 field-collected and 66 generated by adding cropped portions of multiple diseased areas together) with the MobileNetV2 model simulated on a Xiaomi Redmi Note 8 smartphone. The result for the images having multiple disease conditions is described in Table 7. Even in the worst cases, the confidence never reduced below 10%. Hence, the models' confidence threshold was set to 10%. Confidence of classes over 10% was reported to the user as a disease occurrence. Accordingly, the model showed potential in such conditions, where multiple diseases can be present in the ROI.

In sum, this android application can be used for real-time field-level surveys at frequently affected locations and remote regions of developing countries. Since the application does not need an internet connection for work, field surveys can be easier for village-level agronomists and young farmers. Furthermore, the logs collected in the server can be used to monitor the plant disease severity and provide

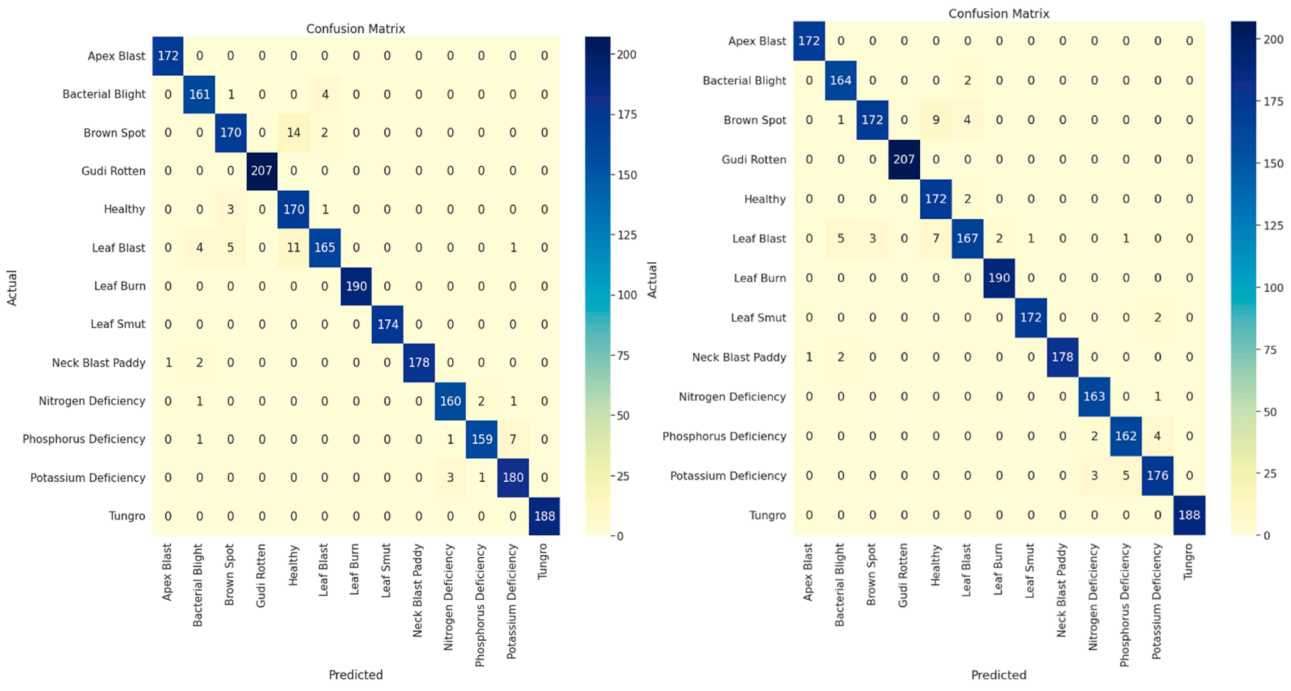


Fig. 5. Confusion matrix of a) ResNet50 model and b) MobileNet V2 model.

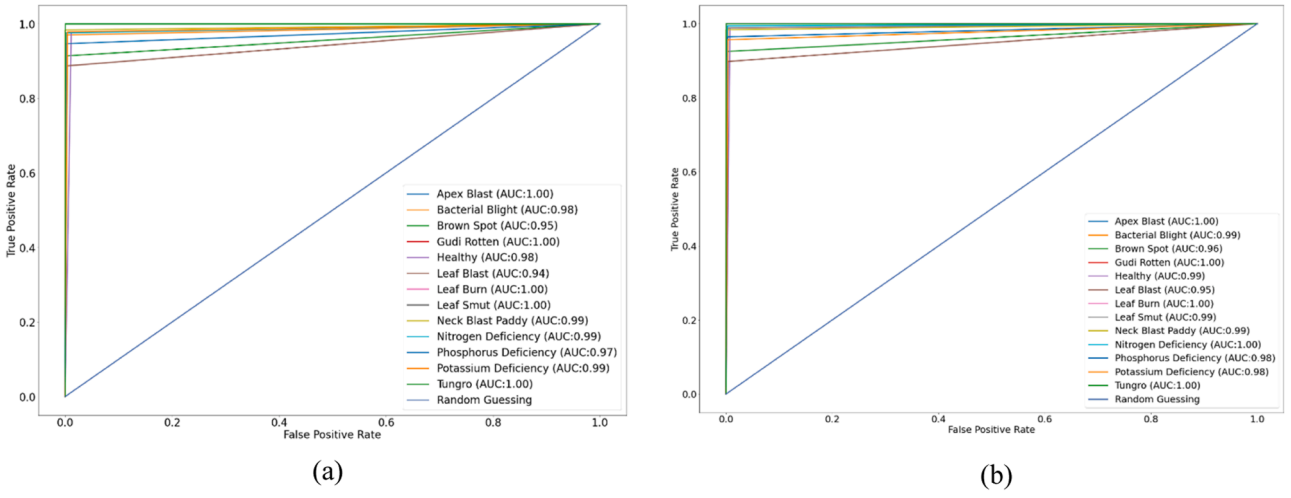


Fig. 6. ROC-AUC curves of a) ResNet50 model and b) MobileNet V2 model.

agricultural advisories in an area by local bodies and research centers.

Besides, this approach can be extended to develop models for other valuable crops with practical application algorithms and model parameter modifications. Currently, the application is more focused on the plant diseases that affect the leaf area; hence the spot segmentation algorithm can be improved to detect the disease spots on other affected plant parts. The Neural networks used have not been trained to detect the micronutrient deficiency symptoms due to a lack of sufficient images, which can be explored in the future. For lab-based rapid disease detection, the application can be configured with a cloud transfer learning pipeline for building a more robust model. Finally, more research is warranted to test the application for smartphones with various hardware configurations.

4. Conclusions

This research evaluated the performance of 11 popular CNN models using stochastic depth optimization and freezing convolution for classifying rice diseases and deficiency symptoms via smartphone-captured images. The ResNet50 model performed best for cloud architectures, while MobileNetV2 was the best model for the smartphone. Finally, the android application “Rice Disease Detector”, compiled with the MobileNetV2 model, performed satisfactorily in identifying multiple disease occurrences in a single capture, highlighting the potential of the proposed approach for future rapid and on-field rice disease detection. This study addressed the on-field detection of complexities like multiple crop disease occurrences, macro-nutrient deficiencies, and diseases existing with nutrient deficiencies. Notably, the selection of the best model for the application development was based on its fast-prediction time, the

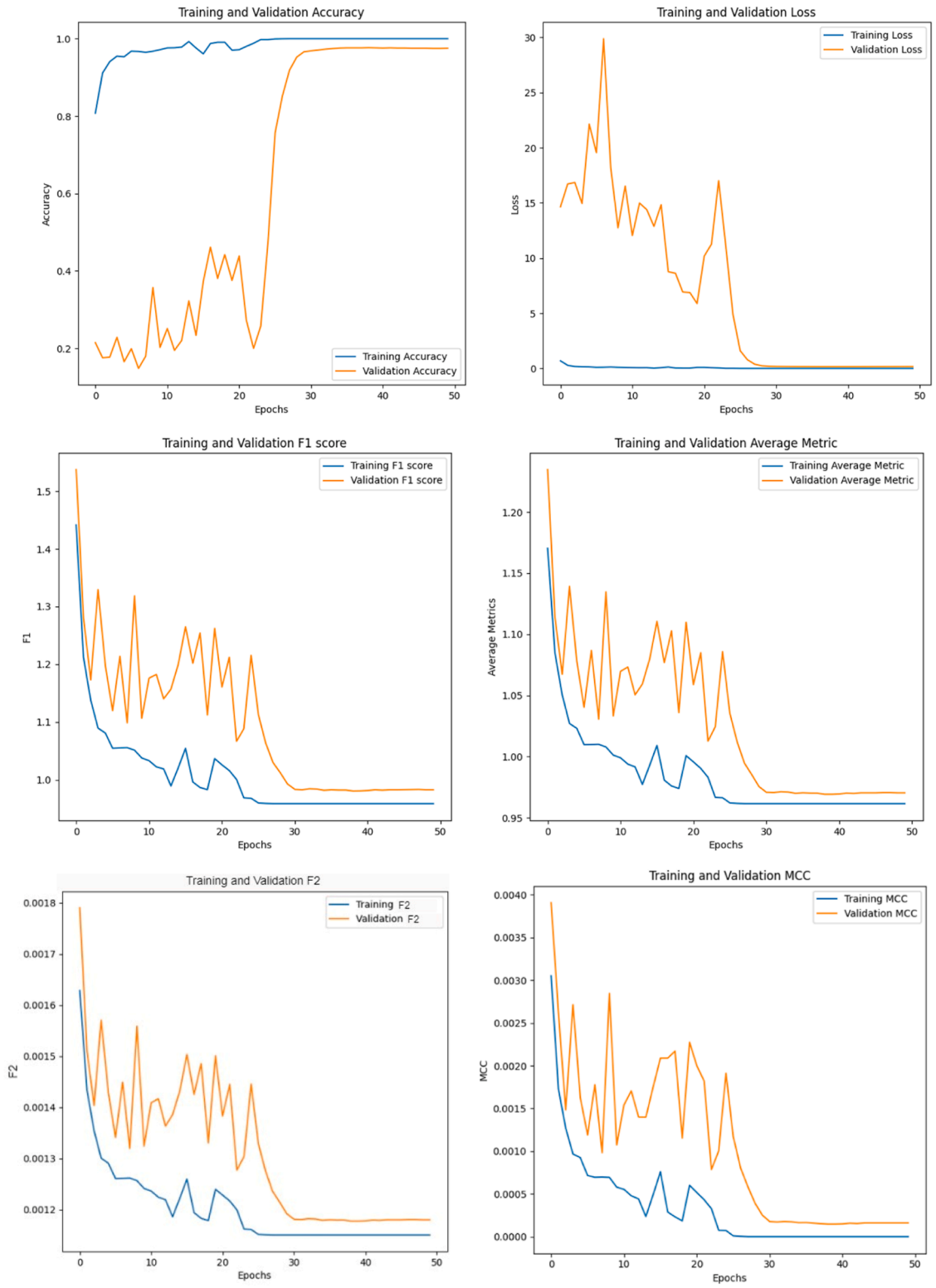


Fig. 7. Performance metrics of the MobileNet V2 model.

capability of handling a large image dataset, and its relatively small size, which is suitable for most of the smartphones available to the farmers. Taken collectively, this study provided a new idea for crop disease detection using average android smartphones among rural populations and field-level advisors, indicating both theoretical and practical significance for disease classification. In the future, the approach proposed herein can be tested to identify crop micronutrient deficiency symptoms that are often neglected.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table 7
Classification results of images having multiple rice diseases.

Class combinations	Number of images under validation	Number of images predicted correctly	Minimum % confidence	Class belongs to minimum confidence	Accuracy
$A + L +$ D	6	6	19	A	1.00
$A + I$	9	7	34	A	0.78
$B + K + G + H$	5	5	26	K	1.00
A + C + L	4	3	23	L	0.75
$D +$ I	10	7	39	D	0.70
$M + K + A$	9	9	35	M	1.00
$F + G + H + K$	5	5	15	H	1.00
$M + N$	8	8	30	N	1.00
C + M + A	11	10	45	A	0.91
$F + K +$ I	2	2	39	I	1.00
$N + D + A + B$	8	7	24	D	0.88
$L + G$	9	6	11	G	0.67
$K + B + N + M$	14	13	43	B	0.93

A = Apex blast, B=Bacterial blight, C= Brown spot, D= Gudi rotten, F= Leaf blast, G= Leaf burn, H= Leaf smut, I= Neck blast paddy, K= Nitrogen deficiency, L= Phosphorus deficiency, M= Potassium deficiency, N=Tungro.

Data availability

Data will be made available on request.

Acknowledgements

The authors wish to thankfully acknowledge financial assistance from the Ministry of Education, Govt. of India.

References

[1] FAO, The Impact of Disasters and Crises on Agriculture and Food Security: 2021, FAO, Rome, Italy, 2021, https://doi.org/10.4060/cb3673en.
[2] T. Deshpande, State of agriculture in India, PRS Legislat. Res. 53 (2017) 6–7.
[3] V.K. Vishnoi, K. Kumar, B. Kumar, Plant disease detection using computational intelligence and image processing, J. Plant Dis. Protect. 128 (2021) 19–53.
[4] A.K. Mahlein, E.C. Oerke, U. Steiner, H.W. Dehne, Recent advances in sensing plant diseases for precision crop protection, Eur. J. Plant Pathol. 133 (2012) 197–209.
[5] M. Sandhu, P. Hadwale, S. Momin, A. Khachane, Plant disease detection using ML and UAV, Int. Res. J. Eng. Technol. 7 (2020) 1–6.
[6] T. Xie, J. Li, C. Yang, Z. Jiang, Y. Chen, L. Guo, J. Zhang, Crop height estimation based on UAV images: methods, errors, and strategies, Comput. Electron. Agric. 185 (2021), 106155.
[7] V. Singh, Varsha, A.K. Misra, Detection of unhealthy region of plant leaves using image processing and genetic algorithm, in: 2015 International Conference on Advances in Computer Engineering and Applications. Presented at the 2015 International Conference on Advances in Computer Engineering and Applications, 2015, pp. 1028–1032, https://doi.org/10.1109/ICACEA.2015.7164858.
[8] Y. Zeng, R. Zhang, T.J. Lim, Wireless communications with unmanned aerial vehicles: opportunities and challenges, IEEE Commun. Mag. 54 (2016) 36–42.
[9] H. Pathak, G. Kumar, S. Mohapatra, B. Gaikwad, J. Rane, Use of Drones in agriculture: Potentials, Problems and Policy Needs, ICAR-National Institute of Abiotic Stress Management, 2020.
[10] J.G.A. Barbedo, L.V. Koenigkan, T.T. Santos, Identifying multiple plant diseases using digital image processing, Biosyst. Eng. 147 (2016) 104–116, https://doi.org/10.1016/j.biosystemseng.2016.03.012.
[11] R. Thabet, R. Mahmoudi, M.H. Bedoui, Image processing on mobile devices: an overview, in: International Image Processing, Applications and Systems Conference. IEEE, 2014, pp. 1–8.
[12] A. Johannes, A. Picon, A. Alvarez-Gila, J. Echazarra, S. Rodriguez-Vaamonde, A. D. Navajas, A. Ortiz-Barredo, Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case, Comput. Electron. Agric. 138 (2017) 200–209, https://doi.org/10.1016/j.compag.2017.04.013.
[13] A. Picon, A. Alvarez-Gila, M. Seitz, A. Ortiz-Barredo, J. Echazarra, A. Johannes, Deep convolutional neural networks for mobile capture device-based crop disease classification in the wild, Comput. Electron. Agric. 161 (2019) 280–290, https://doi.org/10.1016/j.compag.2018.04.002.
[14] Y. Lu, S. Yi, N. Zeng, Y. Liu, Y. Zhang, Identification of rice diseases using deep convolutional neural networks, Neurocomputing 267 (2017) 378–384.
[15] M.J. Hasan, S. Mahbub, M.S. Alom, M.A. Nasim, Rice disease identification and classification by integrating support vector machine with deep convolutional neural network, in: 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), IEEE, 2019, pp. 1–6.
[16] P.K. Sethy, N.K. Barpanda, A.K. Rath, S.K. Behera, Deep feature based rice leaf disease identification using support vector machine, Comput. Electron. Agric. 175 (2020), 105527, https://doi.org/10.1016/j.compag.2020.105527.
[17] A. Cruz, Y. Ampatzidis, R. Pierro, A. Materazzi, A. Panattoni, L. De Bellis, A. Luvisi, Detection of grapevine yellows symptoms in Vitis vinifera L. with artificial intelligence, Comput. Electron. Agriculture 157 (2019) 63–76, https://doi.org/10.1016/j.compag.2018.12.028.
[18] J. Chen, J. Chen, D. Zhang, Y. Sun, Y.A. Nanehkaran, Using deep transfer learning for image-based plant disease identification, Comput. Electron. Agric. 173 (2020), 105393.
[19] U.K. Lopes, J.F. Valiati, Pre-trained convolutional neural networks as feature extractors for tuberculosis detection, Comput. Biol. Med. 89 (2017) 135–143.
[20] N. Petrellis, A smart phone image processing application for plant disease diagnosis, in: 2017 6th International Conference on Modern Circuits and Systems Technologies (MOCAST). IEEE, 2017, pp. 1–4.
[21] Y. Li, J. Zhang, P. Gao, L. Jiang, M. Chen, Grab cut image segmentation based on image region, in: 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), IEEE, 2018, pp. 311–315.
[22] S.A. Shahriar, A.A. Imtiaz, M.B. Hossain, A. Husna, M.N.K. Eaty, Rice blast disease, Annu. Res. Rev. Biol. (2020) 50–64.
[23] G. Jamal-u-ddin Hajano, Q.A. Pathan, A.L. Mubeen, Rice blast-mycoflora, symptomatology and pathogenicity, Sindh Agric. Univ. Tandojam 5 (2011) 53–63.
[24] Z. Xu, X. Guo, A. Zhu, X. He, X. Zhao, Y. Han, R. Subedi, Using deep convolutional Neural Networks for image-based diagnosis of nutrient deficiencies in rice, Comput. Intell. Neurosci. 2020 (2020).
[25] J. Zhang, C.-C.J. Kuo, Region-adaptive texture-aware image resizing, in: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2012, pp. 837–840.
[26] Howe, N.R., Deschamps, A., 2004. Better foreground segmentation through graph cuts. arXiv preprint cs/0401017.
[27] Brock, A., Lim, T., Ritchie, J.M., Weston, N., 2017. Freezeout: accelerate training by progressively freezing layers. arXiv preprint arXiv:1706.04983.
[28] Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K., 2016. Deep Networks with Stochastic Depth (No. arXiv:1603.09382). arXiv. https://doi.org/10.48550/arXiv.1603.09382.
[29] Choi, D., Shallue, C.J., Nado, Z., Lee, J., Maddison, C.J., Dahl, G.E., 2019. On empirical comparisons of optimizers for deep learning. arXiv preprint arXiv:1910.05446.
[30] Hansson, N., Vidhall, T., 2016. Effects on performance and usability for cross-platform application development using React Native.
[31] E. Nowak, F. Jurie, B. Triggs, Sampling strategies for bag-of-features image classification, in: European Conference on Computer Vision, Springer, 2006, pp. 490–503.
[32] Patil, B.M., Burkpalli, V., 2021. A perspective view of cotton leaf image classification using machine learning algorithms using WEKA. Adv. Hum.-Comput. Interact..
[33] A.A. Joshi, B. Jadhav, Monitoring and controlling rice diseases using Image processing techniques, in: 2016 International Conference on Computing, Analytics and Security Trends (CAST), IEEE, 2016, pp. 471–476.
[34] S. Phadikar, J. Sil, A.K. Das, Rice diseases classification using feature selection and rule generation techniques, Comput. Electron. Agric. 90 (2013) 76–85.
[35] Y. Wang, H. Wang, Z. Peng, Rice diseases detection and classification using attention based neural network and bayesian optimization, Expert Syst. Appl. 178 (2021), 114770, https://doi.org/10.1016/j.eswa.2021.114770.
[36] G. Kathiresan, M. Anirudh, M. Nagharjun, R. Karthik, Disease detection in rice leaves using transfer learning techniques, J. Phys. 1911 (2021), 012004, https://doi.org/10.1088/1742-6596/1911/1/012004.
[37] Min, C., Wang, A., Chen, Y., Xu, W., Chen, X., 2018. 2pfpce: two-phase filter pruning based on conditional entropy. arXiv preprint arXiv:1809.02220.
[38] A. Gupta, M. Srivastava, C. Mahanta, Offline handwritten character recognition using neural network, in: 2011 IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE), IEEE, 2011, pp. 102–107.