Analyses of Nine Different Types of Diseases in Paddy with Hybrid Algorithms using Deep Learning

M. Kalai Priya M.E (CSE),
S. Dhanabal
Asp/head of CSE,
PGP College of Engineering and Technology, Namakkal.

***Abstract*:- In India the economic, political and social stability depend directly as well as indirectly on the annual production of rice. The income of hundreds of millions of people depends only on rice production and nothing else. However, as per the report of International Rice Research Institute (IRRI), 37% of the rice yield loss is due to diseases. In this consequence, the farmer can take care of crop on-time with apposite treatment. As a crop, rice is the most significant humanoid nutrition in the world, which can be fed directly than any other harvest. Chronologically, the main objective of farming is to yield and feed food to the nation. So, these leaf diseases in any forms in rice crop tends to cause reduction in quality, yield and fiscal progression respectively. he disease detection and identification in large field through automatic technique is really useful as it reduces the work, time and cost for observation and evaluation of disease symptoms. This project reports a novel approach for detection and identification of rice leaf diseases by using various advanced algorithms such as VGG16, Resnet, etc to increase the accuracy to greater level. To apply loss minimization technique such as categorical cross entropy to increase the accuracy. It increases accuracy with use of hybrid algorithms by performing fine tuning. As well as suggests the type of chemical fertilizer to be used when any type of disease gets detected. As well as suggests the type of chemical fertilizer to be used when any type of disease gets detected.**

Keywords: Rice disease, Pest, Convolutional neural network, Dataset, Memory efficient, Two stage training.

INTRODUCTION:

Timely detection of plant diseases and pests is one of the major challenges in agriculture sector. Rice occupies about 70 percent of the grossed crop area and accounts for 93 percent of total cereal production in Bangladesh [1]. Rice also ensures food security of over half the world population [2]. Researchers have observed 10-15% average yield loss because of 10 major diseases of rice in Bangladesh [3]. So, it is very important to detect the diseases of rice timely for ensuring a sustainable production of rice. Currently, when a rice disease
outbreak occurs somewhere, rice disease specialists of different agriculture research centers or agriculture officials appointed by the government visit the place, and give advice to the farmers. In many areas, there are not adequate rice disease specialists compared to the number of farmers. There is a great need for automatic rice disease detection using easily available devices in rural areas.

Deep learning techniques have shown great promise in image classification. It is a technique based on feature learning from labeled training dataset. In recent years, this technique has been used to analyze diseases of tea [4], apple [5], tomato [6], grapevine, peach, and pear [7].

In most of the cases, they have used leaves or fruits to detect the diseases from the images. In many of these cases, they have used images from homogeneous backgrounds. Moreover, in most cases, the datasets have been crawled from different internet sources. There are some fundamental differences regarding the pattern of diseases between rice plants and the above mentioned plants. First, the diseases and pests of rice plant can affect both stem and grain. Second, the healthy area and the affected area of the rice plants do not have any significant contrast in color. All these factors make it extremely difficult to collect and label the affected rice plants and to recognize the correct disease or pest two studies related to rice disease detection can be found in [8] and [9]. Lu et al .conducted a study on detecting 10 different rice plant diseases using a small handmade CNN architecture inspired by older deep learning frameworks such as LeNet-5 and Alex Net
[8]. They used only 500 images. Some of them were collected from the field. The rest were collected from agricultural pest and insect pests picture database. Though they have reported a high accuracy of 95.48%, We have got only 82% accuracy on our test set using their model. Atole et al. used AlexNet to distinguish among three classes - normal rice plant, diseased rice plant and snail infected rice plant [9]. They used only 227 images taken from rice field.

Our research is the first comprehensive study on rice disease and pest detection using deep convolutional neural networks. In our dataset, there are six different diseases and three different pests. The six diseases are - False Smut, Sheath Blight, Sheath Rot, Bacterial Leaf Blight (BLB), Neck Blast and Brown Spot. The three pests are - Brown Plant Hopper(BPH), Stemborer and Hispa. We have kept Sheath Blight and Sheath Rot disease in the same class, because their treatment method and place of occurrence are the same. We have also created a separate class for healthy plant.

Finally, we have total nine classes - five classes for disease, three classes for pest and one class for healthy plant. Our focus is to build a deep CNN which can recognize if a rice plant is healthy or not, and if not healthy, which disease it

has been suffering from. It can also differentiate between a diseased leaf and a dead leaf. It can detect and recognize diseases on any part of the plant, whether it be leaf, stem or grain. A rice disease may show different symptoms according to weather and soil. Pest attack can show different symptoms according to their stage of attack. We have taken these issues into consideration. These aspects have been described in detail in Subsection 4.3. Our model is expected to do well in real life scenario, becauseour training images were collected in heterogeneous background.

We have used five different types of CNN architectures. They are - VGG16, ResNet50, InceptionV3, InceptionResNetV2 and Xception. For each of them, we have used fine tuning, transfer learning and training from scratch to assess their performance. For all of our architectures, fine tuning the model while training has given the best result. For all three training methods, VGG16 architecture has consistently shown very high accuracy on the test set.

RELATED WORKS:

Many approaches have been applied to correctly identify plant diseases from images. Most of them use general image processing techniques, SVM classifier, K-mean clustering, genetic algorithm [11] etc. Hand engineered features often reflect our limited knowledge about an image, and so, we cannot get more insight. Recently, some researchers are using neural network based approaches in this area. Deep neural networks are much better in disease recognition from images than the traditional image processing techniques.

AUTOMATED RICE DISEASE DETECTION APPROACH:

CNN based classifier was used to detect rice plant anomaly in [9]. There were three classes in total which included normal, diseased and snail infected rice plant. They collected227 images in rice fields from nearby districts. They used transfer learning on AlexNet which is a relatively small scale and old CNN architecture. They used various image augmentation techniques on these collected images and trained their CNN model for ten epochs managing to get test accuracy of 91.23%. During the classification task, this approach did not consider the specific class of disease that may affect rice plant. The model can only tell if the plant is affected by disease or not. The dataset contained only 227 images. So, it is likely that the model is not generalized to rice plants outside the image collection region. Many better performing CNN architectures have appeared after AlexNet. They were not used in this paper.

AUTOMATED DISEASE DETECTION APPROACH FOR OTHER PLANTS:

In [12], the authors used deep CNN to detect disease from leaves. They trained the neural network with 54306 images of 14 crop species, which represented a total of 26 diseases along with healthy leaves. Though the accuracy was 99.35% on held-out test set, the accuracy fell to 31.4% when tested on another verified dataset of 121 images captured in real life scenario. In [7], the authors used Caffe Net model to recognize 13 different types of plant diseases. They considered a total of 15 classes- 13 classes for diseases, one class for healthy leaves and one class for background. The plants included apple, peach, grapevine, pear etc. They collected all the images for training from the Internet. Neural network ensemble (NNE) has also been used to recognize five different diseases of tea plant from tea leaves[4].

The authors achieved an accuracy of 91% on the test set. The dataset consisted of only50 images, 10 images of each class. A feed forward back propagation neural network was built from scratch in [13] to detect the species of plant from leaf image. The percentage of leaf affected by diseases or pests was also detected. The neural network did not classify different diseases. The interesting part is that no convolutional neural network was used here. Image edges were detected using Prewitt edge detection algorithm. Some authors used visual spectrograph in ambient lighting conditions to detect yellow rust infestation on winter wheat [14]. They investigated the difference in spectral reflectance between healthy and diseased wheat plants at an early stage in the development of the disease.

PLANT DISEASE STAGE DETECTION AND DISEASE FORECASTING:

Recently some works have been done in recognizing the stage of the disease. In [5], the authors detected four severity stages (healthy stage, early stage, middle stage, and end stage) of apple black rot disease using PlantVillage dataset. They used two different types of training methods: training small convolutional neural networks of different depths from scratch, and fine tuning four state-of-the-art deep learning models such as VGG16, VGG19, Inception-v3, and ResNet50. The best model they found was the fine-tuned VGG16 model, which achieved an accuracy of 90.4% on the test set. Bhagawati et al. trained a neural network with weather parameters such as temperature, relative humidity, rainfall and wind speed to forecast rice blast disease [15]. The prediction accuracy of the model was found to be between 81-87% when data of the same site were used. But they did not test the accuracy of the model in other sites.

PLANT DISEASE LOCALIZATION:

A real time tomato plant disease detector was built using deep learning in [6]. They considered simultaneous occurrence of multiple diseases and pests, and they also considered different infected areas like stem, leaves, fruits etc. They also collected images of different stages of the same disease. The dataset consisted of about 5000 images collected from different farms of Korea. Most of these images had heterogeneous background. Several geometric and intensity transformations were used to increase the number of images. The authors used three main families of detectors: Faster Region-based Convolutional Neural Network (Faster R-CNN), Region-based Fully Convolutional Network (R-FCN), and Single Shot Multibox Detector (SSD), which they considered as “deep learning meta-architectures”. Their models both recognized and localized nine different diseases and pests with the best accuracy of 85.98%.

CHALLENGES:

There are many challenges in correctly identifying rice diseases and pests. We have overcome the limitations of the previous works in this field. We have also kept the challenges mentioned in [16] in our mind while using CNN architectures for classification.

IMAGE COLLECTION:

People working in the agriculture institutes and agriculture universities of Bangladesh do not generally collect images of rice diseases and pests except for presentation and demonstration purposes in a very small scale. We had to collect them from the paddy fields. While collecting images, we have tried to capture images in various capture conditions. We have taken images in windy, sunny and rainy weather. We have also captured images of diseased parts in both summer and winter.

This has helped with training the model in away that it can do well in real life scenario in any possible weather. Heterogeneous back- ground is an indispensable feature of any real life image. We have considered the presence of human, colored sheet, rice field, human body part and many other possible backgrounds while capturing our desired disease and pest images. Our models have achieved a very high classification accuracy on images captured in realistic condition.

INCREASING CLASSIFICATION ACCURACY:

When someone tries to capture the image of diseased area of a rice plant in a rice field, he is likely to capture an image with a background composed of other rice plants, soil, humans, and many other objects. Heterogeneous background makes it quite difficult to segment the region of interest. The symptoms of many rice diseases do not have any well defined boundary. Rather, the color of the diseased area gradually fades into the healthy part of the plant. As a result, image segmentation before using neural network becomes almost impossible. We have tried to keep the number of classes as small as possible using in depth agriculture related knowledge. We have used various image augmentation techniques on our training set to increase accuracy. We have successfully used small CNN models with good accuracy in rice disease detection using techniques inspired from fine tuning.

TRAINING OUR MODEL:

Each of the CNN architectures we have used has a very large number of trainable parameters. For example, VGG16 has 138 million parameters. Training these CNN architectures from scratch or fine tuning these architectures take a lot of time in non-GPU environments.

We have used a remote server to get fairly good amount of GPU which has benefited us with much smaller training time. The GPU instance is shared.

REDUCING MODEL SIZE:

Because of lack of internet connectivity in rural areas of developing countries, CNN models need to run offline in rice disease and pest detection oriented mobile applications. As we reduce the number of parameters in CNN models, their classification capability decreases, which reduces their utility as a tool for disease and pest classification module. We introduced stacked CNN architecture based on two stage training keeping this memory accuracy trade- off in mind.

DATA COLLECTION:

Rice diseases and pests occur in different parts of the rice plant. Their Occurrence depends on many factors, such as temperature, humidity, rainfall, variety of rice plant, season, nutrition etc. So, the task of data collection in field level is a lengthy and challenging task.

CLASSES CONSIDERED:

We have a total of five classes for disease, three classes for pest and one class for healthy plant. Symptoms of different diseases are seen at different parts of the rice plant such as leaf, stem and grain. Bacterial Leaf Blight disease, Brown Spot disease, Brown Plant Hopper pest (at its late stage) and Hispa pest occur on rice leaf. Sheath Blight disease, Sheath Rot disease, Brown Plant Hopper pest (at its early stage) occur on rice stem. Neck Blast disease and False Smut disease occur on rice grain. Stemborer pest occurs on both rice stem and rice grain. So, we have considered all these parts while capturing images.

To prevent our model from being confused between dead parts and diseased parts of rice plant, we have collected enough images of dead leaf, dead stem and dead grain of rice plants. Images of the dead parts of the plant are considered in the class of healthy plant. We consider a total of nine classes. A sample image of each rice disease and pest has been provided in Figure 1. Sheath Blight, Sheath Rot and their simultaneous occurrence have been considered in the same class, because their treatment method and place of occurrence are the same.

QUANTITY OF IMAGE DATA:

We have captured 1426 images of rice plants infected with diseases and pests along with healthy rice plant from the field of BRRI from December, 2017 to June, 2018 for a total of seven months. The total number of images of these different classes that we have collected is shown in Table 1. Some major pests such as Gall Midge and Leaf Roller are not included in this study as we could not get enough of these pests in the field. Moreover, Ufra and Leaf Blast disease occurred in small scale in rice field during our study. There were very small amount of these diseases in the rice fields during our data collection period. But Image capture conditions and image backgrounds from field are certainly different from images captured in the nursery. The low amount of data is not sufficient to train a neural network.

We have not included these two diseases.

VARIATION IN SAME CLASS DATA:

We have considered necessary variations while collecting data from the field. The more the variation in the data set, the better is the generalization of the trained model. This means that a model trained with a data set with a lot of variations will be able to generalize and perform well on test set.

Four different types of camer have been used in capturingthe images. The images have been captured in the rice field in real life scenario. We have captured images in different types of backgrounds. In some images, the background is the surroundings of the field, and in some other images, the background is our hand or papers of different colors. This makes our model robust to any change in background. Weather conditions are also different at different times. Some images have been captured in overcast conditions, some have been captured in sunny weather. False Smut, Stemborer, Healthy Plant class, Sheath Blight and/or Sheath Rot class have multiple types of symptoms. We have covered all the symptoms of these classes. Moreover, early stage symptoms of Hispa and Brown Plant Hopper are different from their later stage symptoms. We have also covered these aspects while collecting data of these classes.

AGRICULTURE SPECIfiC KNOWLEDGE:

We have used in depth knowledge of agriculture in order to reduce the number of classes for better accuracy. Brown Spot disease often occurs with other diseases in the same plant. We have considered only the disease occurring with Brown Spot as a separate class, because when a particular disease occurs with Brown Spot disease, treatment for that particular disease is applied only. No separate treatment is applied for Brown Spot disease. Treatment for Brown Spot is applied only when it occurs as a single disease. We have also kept Sheath Blight, Sheath Rot and their simultaneous occurrence in the same class. Symptoms of Sheath Blight and Sheath Rot look quite similar in many cases and sometimes they are seen to occur together. These two diseases attack the same organ of rice plant (rice stem), and have the same treatment method. So, they have been kept in the same class. The different stages of the diseases and pests which we have considered do not differ in terms of treatment method. So, we have not created different classes for different stages of attack of the same disease or pest.

IMAGE PROCESSING TECHNIQUES:

A lot of images are needed for each class in order to train the CNN model. We have used a random mixture of the various image augmentation techniques in order to create eight images from each of our captured image. We have associated a probability with each image augmentation technique. Based on the probability, a particular technique is used or is not used while generating a particular augmented image.

We have used random rotation from -15 degree to 15 degree. We have used rotations of multiple of 90 degree at random. CNN classification is not rotation invariant in general. So, these two transformations are of high importance, and they are assigned a high probability. Other transformations such as random distortion, shear transform, vertical flip, horizontal flip and skewing have also been used. Every augmented image is the result of a particular subset of all these transformations. The results of the combinations of these geometric transformations are shown in Figure 9. We have also generated two more images from each image of different intensity using intensity transformation shown in Figure 8. This kind of variation in training data helps the model to actually learn the features of each disease rather than just memorizing training set examples for each class. Memorization of training data causes overfitting and leads to poor test set accuracy.

OUR SOLUTION:

Our work aims to effectively detect eight classes of diseases and pests that affect rice plants along with healthy rice plants using CNN. We have collected a large dataset from the field of BRRI by capturing images of rice plants infected with diseases and pests in real life scenario described in Section 4. We have a total of nine classes- five classes for diseases, three classes for pests and one class for healthy plant. We annotate the images to train our CNN architectures by putting the images of different classes in separate folders. We randomly pick 70% of all the images of each class and put them into training set. Similarly, another 15% of the images of each class are put into the validation set and the rest of the images are put into the test set. The intersection of training set, validation set and test set is empty. Next we increase the number of images in the training set ten times using different image augmentation techniques and intensity transformations.

OUR APPROACH USING STATE-OF-THE-ART CNN ARCHITECTURES:

We use Keras framework with tensorflow backend to train our models with our training set. After each epoch of training, we evaluate the performance of the model on our validation set. If the validation accuracy exceeds the best validation accuracy we get so far, we save this model. In this way, we always save the model with the best validation accuracy so far during training. After training for 150 epochs, we stop training and evaluate the performance of the model on our test set.

We use some modern CNN architectures such as VGG16, InceptionV3, ResNet50, Xception and InceptionResnetV2. Each of the architectures has some unique charac- teristics. VGG16 [10] is a sequential convolutional neural network using 3×3 convolutionfilters. After each maxpool layer, the number of convolution filters gets doubled in VGG16. InceptionV3 [17] is consisted of inception blocks. In each inception block, convolution filters of various dimensions and pooling are used on the input in parallel. They are concatenated along their channels just before providing output. Resnet50 [18] is a very deep convolutional neural network with skip connections from earlier layer apart from the direct connection from the immediate previous layer. InceptionResnetV2 [19] combines the notion of parallelism of Inception architecture and skip connection of Resnet architecture. Xception [20] is the result of depthwise separable convolution which implies that spatial convolution and cross channel convolution are completely separated. The number of parameters of each of the five state-of-the-art CNN architectures that we have used have been given in Table 3. We use three variations of training methods for each of the architectures.

Baseline training: In this method, we train all the network layers from scratch. We randomly initialize all the layers and train them from scratch. This method of training takes a lot of time to converge but produces fairly good accuracy. We denote this training method as B.

Fine Tuning: In this training method, we keep the pre-trained image net weights of the convolution layers in tact. We randomly initialize the weights of densely connected layers only. Then we train all the layers until convergence. It is to note that the convolution layers are trained from their pre-trained image net weights, and the dense layers are trained from randomly initialized weights. We denote this method as FT.

Transfer Learning: In this method, we do not train the convolution layers of the CNN architectures at all. Rather we keep the pre-trained imagenet weights. We only train the dense layers from their randomly initialized weights. We denote this method as TL.

We resize all the images of our dataset to the default image size of each architecture before working with that architecture. This makes training and validation step in each epoch faster compared to run-time resizing. For example, we resize all the images of our training, validation and test set to 299×299 pixel size before working with architectures such as Xception, InceptionV3 and InceptionResNetV2. Similarly, for other models, all the images are resized to 224×224 pixel size.

OUR APPROACH WITH MEMORY EffiCIENT STACKED CNN:

We make two CNN architectures from scratch. The building structure of these two CNN models have been inspired from the sequential nature of VGG16, which has given the best accuracy on our test set. Both the models are given in Figure 10. The model on the left has 2 million parameters and the model on the right has only 0.8 million parameters compared to 138 million parameters of VGG16. We follow two approaches for training these two models.

Baseline Training: We randomly initialize the weights of all filters and dense layer nodes and train from scratch using our dataset of original nine classes. We use the validation set to tune the hyperparamters and finally, we use the test set to report test accuracy.

Two Stage Training: In the first stage, we divide our entire image dataset of nine classes into 17 classes by keeping all intra-class variations in separate classes. These variations have been shown in detail in Subsection 4.3. For example, we divide the healthy plant class into three separate classes because of intra-class variation. Now, we train the model with this 17 class dataset. As a result, the final dense layer of our model has 17 nodes with softmax activation function. We use the validation set to tune the hyperparameters. This validation set also has 17 classes. We do not use any test set in the first stage. In the second stage, we use the original dataset of nine classes. We now have a trained model from the first stage of training. We keep all layers and their corresponding weights in tact except for the topmost layer. We replace this layer of 17 nodes with a layer of nine nodes with softmax activation function in order to match the current nine classes in our stage two training data. The validation set also has nine classes and is used to tune hyperparameters. In this second stage, we have a test set of nine classes. We test our model after stage two training and hyperparamete tuning, and report the test accuracy. So, the concept behind the name stacked CNN is that after first stage is over, we pop the topmost layer of our CNN architecture and push a new layer on top with different number of nodes for the second stage. We have seen a significant increase of accuracy with our small models using this special two stage training method. This training method has been inspired from fine tuning, which is one of the three types of training methods that we have used for our five state-of-the-art CNN architectures.

EXPERIMENTAL SETUP:

In experimental setup, we determine performance evaluation metric of the CNN architectures, form their general structure, identify hyperparameters, tune those hyperparameters and set up environment for conducting experiment.

Performance Metric: We use accuracy as our performance metric. We save the model weights after the epoch in which we have the best validation accuracy. In our data set, number of samples is fairly distributed among the nine classes. Accuracy is a good performance measure when number of samples is not biased towards any particular class.

Loss Function: We use Categorical Crossentropy as our loss function. We have total ten classes. This loss function is actually a log loss version of multiclass case.

Basic Structure Adapted for State-of-the-art CNN Architectures: We use several CNN architectures such as VGG16, InceptionV3 etc. We remove the top three layers of the original convolutional neural network architectures. We flatten the output of the last layer among the remaining layers of the architectures. The remaining convolutional layers carry the pre-trained weights from imagenet classification task. As a result, these architectures are already well-acquainted with basic image features. We add three densely connected layers with relu activation function and one dense layer with softmax activation function on top. This topmost layer with softmax activation function contains nine nodes for the nine classes.

Hyperparameter Tuning: Hyperparameters are not trainable. We fix their values at the start of training. We consider validation accuracy while tuning hyperparameters. We tune two hyperparameters. Dropout rate is the first hyperparameter that we tune. A dropout rate of 0.3 means that our model will ignore 30% of the neurons of the previous layer at random. It helps to reduce overfitting. We add dropout layer after each dense layer except for the last layer. We test our models with dropout rate of 0.3, 0.4 and

0.5. Dropout rate of 0.3 has given the best result in general. The second hyperparameter that we tuneis Learning rate. Learning rate determines how fast our model weights are adjusted in order to get to the local or global minima of our loss function. We have tested our model with learning rate of 0.01, 0.001, 0.0001 and 0.00001. In terms of convergence speed and accuracy, learning rate of 0.0001 has given the best result.

Optimizer: We use Adaptive Moment Estimation (Adam) for training our models. It is a method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients, Adam also keeps an exponentially decaying average of past gradients. It combines the advantages of two other extensions of stochastic gradient descent - AdaGrad and RMSProp.

Experimental Environment: We use a remote Red Hat Enterprise Linux Server of RMIT University for training our convolutional neural network architectures. The processor is Intel (R) Xeon (R) CPU E5-2690, whose clock speed is 2.60 GHz. It has 56 CPUs with two threads per core. There is 503 GB RAM available to us. Each user can use up to 1 petabyte of storage. There are also two GPUs available. Both of them are 16 GB NVIDIA Tesla P100-PCIE GPU.

Experimental Evaluation:
We have trained our dataset with five state-of-the-art CNN architectures. They are -VGG16, ResNet50, InceptionV3, InceptionResNetV2, Xception. The performance of these CNN architectures for fine tuning (FT), transfer learning (TL) and training from scratch(B) is shown in Table .

To compare our approach with an existing approach [8], we trained the model used in [8] with our dataset. If we set image size to 224 × 224, we get 87% accuracy on test set. The image size mentioned in [8] is 512 × 512. This image size gives validation and test accuracy below 30%.

Here we see that all of our architectures have given their best accuracy on test set when we fine tune them from pre-trained imagenet weights. Moreover, test accuracy of the architectures when trained from scratch is comparable to the test accuracy of corresponding architectures when fine tuning them from imagenet weights. In fine tuning, we start from imagenet pre-trained weights and then train the whole network on our dataset. So, it is easy to reach the global optimum. On the other hand, we do not train the layers of the original convolutional architecture in transfer learning. So, the model may not capture all the characteristics of the dataset. That is why, accuracy in transfer learning has been found to be lower compared to the other two training methods. For example, InceptionV3 has given test accuracy of only 82% when using transfer learning while its test accuracy when using fine tuning was 89.1%. ResNet50 has failed to capture the characteristics of our dataset when we apply transfer learning on it. It has achieved only 24.8% test accuracy while applying this training method. In general, architectures with skip connections (ResNet50, Xception) have shown bad performance on the test set when we have applied transfer learning on them compared to other models with no skip connection. Both the validation accuracy and the test accuracy are very high for the best performing variation of each architecture, i.e. when we fine tune an architecture. As we use dropout in the dense layers which we have added, there is very little chance of overfitting. For example, fine tuned ResNet50 and Xception have given high test accuracy of 97.5% and 98.1% respectively. Our training, validation and test set come from the same distribution. This may be a reason for such good performance. From the results in Table 4, we see that fine tuned VGG16 model has performed best, achieving an accuracy of 99.53% on the test set, although its number of parameters is much higher than the other four architectures

In Figure 10, we can see that we made two CNN architectures from scratch. We name the CNN with 2 million parameters as model A and the CNN with 0.8 million parameters as model B. The performance of these CNN architectures for Baseline training and two stage training is shown in Table 5. Although model B is very small and has a poor test accuracy of only 79.2% with baseline training, it achieves 91.7% test accuracy in two stage training model A in spite of being much smaller than state-of-the-art CNN architectures achieves test accuracy of nearly 95% after going through two stage training. The reason for such good accuracy is that in the first stage of two stage training, we make the model learn not only features related to inter-class variation but also features related to intra-class variation. As a result, in the second and final stage of training, the model is able to discriminate better among the original classes. The drawback of the method is that one has to perform a lot of manual intra-class variation related data labeling for data preparation in the first stage of training. It is to note that the models that we build are sequential in nature. Introducing non-sequential aspects such as skip connection, parallel branching, depth-wise separable convolution in building our model, we can achieve better accuracy with even smaller models by implementing two stage training on our dataset. We leave it for future work.

We have generated graphs of accuracy vs epoch number and loss vs epoch number for all five state-of-the-art CNN architectures. These graphs are given in Figure 11. Here, FT stands for fine-tuning the weights from imagenet pre-trained weights. We have used batches for training. Batch is a bundle of training samples used to update model weights all together. That is why our accuracy and loss fluctuate a bit for all
the architectures.

We generate confusion matrix for each variation of each architecture on the validation set and test set. A confusion matrix gives a quantitative representation of each class of image being misclassified as an image of another class. A sample confusion matrix is provided in Figure 12. If the number of misclassifications between two particular classes become high, it indicates that we need to collect more data on those classes to properly train the CNN architecture so

that it can differentiate between those two classes. For this purpose, we generate confusion matrix on our validation set for each CNN architecture. There were a lot of confusions between Healthy Plant class and Brown Spot class. By collecting more data on these two classes, we have eliminated those confusions.

Figure 12 shows the confusion matrix of VGG16 (FT) on the test set. We see from the figure that Bacterial Leaf Blight disease has not been misclassified with any disease or pest. On the other hand, one image of Brown Plant Hopper has been misclassified with one image of Healthy Plant class. BLB represents Bacterial Leaf Blight, BPH represents Brown Plant Hopper, BS represents Brown Spot, FS represents False Smut, NB represents Neck Blast, SBR represents Sheath Blight and/or Sheath Rot in Figure 12.

CONCLUSION:

We have proposed deep CNN based classifier for real time rice disease and pest recognition. We have also introduced two stage training for achieving desired accuracy with our memory efficient stacked CNN architecture. We have conducted a comprehensive study on rice disease and pest recognition, incorporating nine classes of rice diseases, pests and healthy plant. We have collected a lot of images of rice plants infected with various diseases and pests from the rice field in real life scenario.

We have applied the knowledge of agriculture in solving rice disease classification problem. We have implemented various training methods on each of the CNN architectures. We have successfully been able to distinguish between inter-class and intra-class variations of diseases and pests in rice plant in complex environment. Validation accuracy and test accuracy of most of the CNN architectures are found to be very high, because our training, validation and test set have been collected from she same site. We plan on incorporating location, weather and soil data along with the image of the diseased part of the plant to develop a comprehensive and automated plant disease detection mechanism. We plan to work with memory efficient non-sequential CNN models inspired from ResNet, Inception and Xception architecture in order to achieve higher accuracy in plant disease and pest classification in future. This will facilitate automated and accurate disease detection using mobile devices.

ACKNOWLEDGMENTS:

We thank the authority of BRRI (Bangladesh Rice Research Institute) for supporting the research by providing us with the opportunity of collecting a lot of images of rice plant diseases in real life scenario. We also thank RMIT University for giving us the opportunity to use their GPU server.

REFERENCES:

[1] T. Coelli, S. Rahman, C. Thirtle, Technical, allocative, cost and scale efficiencies in bangladesh rice cultivation: A non-parametric approach, Journal of Agricultural Economics 53 (3) (2002) 607–626.

[2] Part ii - rice in world trade, http://www.fao.org/docrep/006/Y4751E/y4751e03.htm, accessed: 2018-10-14.

[3] Some appropriate technologies of rice cultivation (bengali bulletin).

[4] B. C. Karmokar, M. S. Ullah, M. K. Siddiquee, K. M. R. Alam, Tea leaf diseases recognition using neural network ensemble, International Journal of Computer Applications 114 (17).

[5] G. Wang, Y. Sun, J. Wang, Automatic image-based plant disease severity estimation using deep learning, Computational intelligence and neuroscience 2017.

[6] A. Fuentes, S. Yoon, S. C. Kim, D. S. Park, A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition, Sensors 17 (9) (2017) 2022.