International Journal of Innovative Research in Engineering & Management (IJIREM)
ISSN: 2350-0557, Volume-8, Issue-6, November 2021
https://doi.org/10.55524/ijirem.2021.8.6.16
www.ijirem.org

A Novel Design of Rice Leaf Disease Detection Model Using Machine Learning

Taha Hussain
M.Tech Student,
RIMT University.
Mandi Gobindgarh,
Punjab, India

Dr. Jasmeen Gill
Associate Professor,
Department of Computer Science
& Engineering,
RIMT University, Mandi-
Gobindgarh Punjab, India

Ravinder Pal Singh
Technical Head, RIMT-DRI
RIMT University
Mandi Gobindgarh,
Punjab, India

ABSTRACT

Rice has a high nutritional value since it includes several vital elements. It is one of the most widely consumed foods. However, due to the great diversity of rice, judging its quality is difficult. The use of photography and machine learning in this work resulted in a unique method for determining the quality of rice without causing any damage or loss. First, a DSLR camera was used to capture pictures of the Rice leaf samples. The data was separated into healthy. and sick groups and sorted. The CNN was then used to discriminate between different types of rice leaf diseases. Various parameters have been used to train several models**.** Here we use graphical user interface(GUI) as an interface software. Five classes of leaves ie. Healthy, Bacterial leaf blight, Brown spot, leaf smut, and leaf blast were taken into account, and all the leaves where classified into one of these classes. Finally, the models were evaluated based on the outcomes of the experiments. The finest performance was noticed by CNN. CNN's classification and average cost time accuracy were 95 percent and 0.01 seconds, respectively. Overall, the results demonstrate that picture data generated by the system may be utilized to assess rice quality quickly, accurately, and safely.

Keywords

Rice, Photography, CNN, detection, Quality, Image data

1. INTRODUCTION

Rice yields about 497 metric tonnes per year across the world. Rice is very important in the rural economy. India produces about a quarter of all rice in the globe. In India, rice is a basic meal. Rice is exported in large quantities from India to other areas of the world [1]. The primary reason for this is because cultivating a variety of agricultural genotypes has a significant influence. As a result, there is a broad range in quality, sizes, weights, and features [2]. Despite the fact that new species are being offered as a solution, the grading process may be an interesting subject to research due to the growing number of farms with rice fields. The neural graders created by computers to imitate human judgements about product quality were thoroughly investigated [3].

It has never been more important to rectify, promptly detect diseases, including earlier prevention, in our changing environment [4]. Pathologies can be identified in a variety of ways. Some illnesses have no obvious symptoms, or the effect is too early to intervene, necessitating a comprehensive examination. Nonetheless, most illnesses have obvious symptoms, necessitating a certified naked eye inspection that is adequately trained and used to diagnose plant disease in practice [5]. A plant pathologist must have an excellent observer ability to recognize frequent symptoms in order to make an accurate diagnosis of plant disease [6].

Computer Vision Progress [7] gives a chance for precise plant protection practices [8] to develop, improve, and expand the precision agricultural sector.

ANN is a machine learning and cognitive research paradigm for information processing inspired by biological nervous systems and process information, including intellectual nervous systems. The brain is made up of many neurons that are tightly connected and work together to solve issues.

The neuron's output is neither 0 nor 1. Instead, it's known as the transfer function. There are many different forms of transfer features: step, linear, sigmoid, and so on. Because of this flexibility, even little weight and bias changes result in minor neuronal performance differences.

$$\Delta\text{output} \approx \sum \frac{\partial\,\text{output}}{\partial w_j}\Delta w_j + \sum \frac{\partial\,\text{output}}{\partial b}\Delta b. \qquad (1)$$

Basically, the small change in weight or bias causes the small corresponding change in the network output (Figure 1).

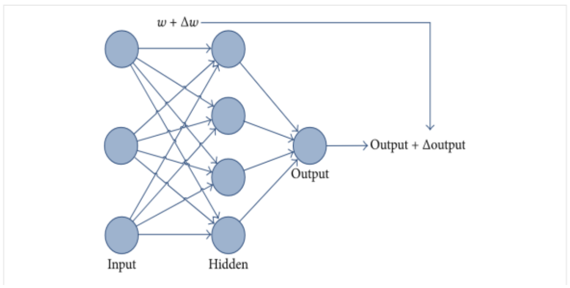


Figure 1: Small change in weigh causing small changes in Output

In the subject of design, there are two types of ANNs: ANN feeds, in which each layer's output is unlikely to affect the same layer, and ANN feedback, in which signals flow in both ways via network loops.

To identify and categorise plant diseases, traditional digital image transformation methods such as colour analysis and thresholding[9] have been employed. Plant diseases currently use a variety of methodologies, the most common of which being artificial neural networks[10] and SMVs[11]. They're coupled with a variety of picture preprocessing techniques to improve feature extraction.

Here in this model, five classes [11] of leaves are considered and all the datasets are classified as one of these five classes.

The five classes are as under:

- **Healthy:** The leaves are green , disease free , spot free and fresh. These plants' rice quality is up to the mark and are considered to grow well and are therefore healthy for consumption [12].
- **Bacterial leaf blight:** Bacterial blight is caused by *Xanthomonas oryzae*pv. *oryzae* (Xoo) and affects the rice plant at the seedling stage where infected leaves turn grayish green and roll up [13]. As the disease progresses, leaves turn yellow to straw-colored and wilt, leading whole seedlings to dry up and die. The disease occurs in both tropical and temperate environments, particularly in irrigated and rainfed lowland areas [14]. It is commonly observed when strong winds and continuous heavy rains occur. The disease is severe in susceptible rice varieties that are treated with high nitrogen fertilizer [15].
- **Brown spot:** Dark brown colored and round to oval shaped lesions on rice leaves destructive for the quality of the rice yielded.
- **Leaf smut:** Small black linear lesions on leaf blades, leaf tips may turn grey and dry [16].
- **Leaf blast**: Rice blast (*Pyricularia grisea*) is a fungus that feeds on the rice plant, causing severe damage usually during the seedling stage. It attacks different parts of the plant [17]: the collar, which can ultimately kill the entire leaf blade; the stem, which turns blackish and breaks easily (node blast); the neck of the panicle, where the infected part is girdled by a grayish brown lesion [18], or when severe, causes the panicles to fall over; or on the branches of the panicles which exhibit brown lesions when infected.

The study employs a well-trained neural deep convolution network [19] that has been particularly tailored to a plant leaf database obtained for various plant illnesses, as well as a new approach for identifying plant ailments. The model is fresh and advanced: healthy feed and backdrop pictures are consistent with other classes, allowing the model to differentiate between healthy and ill plants or the surroundings using deep CNN.

2. LITERATURE REVIEW

The texture extraction feature may also be utilised to figure out how healthy a plant is. Patil and Kumar[19] presented a model for diagnosing plant disease based on textural characteristics such as inertia, homogeneity, and the correlation of the grey matrix in the image. The corn leaves were tested for infections in combination with the colour extraction.

With all of these aspects, picture quality improves and categorization improves. The authors investigated the well-known traditional approaches of feature extraction in [20].

Because artificial intelligence sciences are constantly evolving, the study in this study is largely focused on the application of these techniques (AI).

In the feedback spread of the neural network, certain techniques are used to identify leaf species, pest species, or illnesses, including input, output, and one hidden layer [21]; the authors propose this model. They've created software that can detect pesticides or illnesses in farms and fix them.

The authors use characteristics from the Particle Swarm Optimization (PSO)[23] and forward neural networks in[22] to detect the areas of the injured cotton floor and improve systems precision by a final 95 percent overall accuracy..

Plant diseases can be diagnosed and distinguished using vector machinery aid techniques. The method was used to categorise sugar beet diseases[24], with accuracy ranging from 65 percent to 90 percent depending on the kind and stage of the illness.

3. METHODOLOGY

It explains the complete process of building the model for detecting deep CNN plant illnesses. The entire process, which began with the collection of images for categorization using deep neural networks, has been broken down into various phases in the subsections below.

3.1. Dataset

From the training phase through the performance evaluation of recognition algorithms, appropriate data sets are necessary at all stages of object identification research. The DSLR camera was used to take all of the photos for the dataset. In the dataset, the pictures were divided into two groups. The term 'dysfunction' was used to describe a class. A second category has been dubbed "healthy."

A second class was created in the data set to distinguish between sick and healthy Rice. It is only exhibited with images of healthy leaves. The addition of a backdrop picture class to the dataset helped generate more accurate classifications..

The next step was to photograph the dataset. The study's main goal is to train the network so that the classes can be comprehended. As a result, as the pictures are extended, the Network's odds of learning the proper characteristics improve. In addition, a database of 2000 training photos and 589 validation photos was generated.

3.2. Data Pre-processing

In general, two preparation steps are required before sending pictures to the network. The photos will first be scaled to fit into the CNN input layer. AlexNet sizes of 227-227 are common, as are DenseNet, ResNet, and VGG sizes of 224-224. Second, pictures must be normalised so that the model can more rapidly and broadly converge unseen data. Additional pre-treatment was considered. The images have been altered using Mohanty's grayscale. Mohanty compared the accuracy of greyscale pictures to that of colour ones. On colour models, the f1 score increased somewhat from 1.34 percent to 3.33 percent.While colour photos aid in illness detection, performance suffers somewhat when converted to grayscale. In the same study, the scientists looked at the impact of removing the backdrop. In reality, one of the most common issues with photography is backdrop control. The analysis is prepared using traditional image processing tools. In the backdrop, Mohanty et al(2016) performs somewhat better, with an improvement of less than 1% in the f1 score. Because in the field, background segmentation is not a possibility, and the CNN's backdrop cannot be removed.

3.3. Training Phase

The intrinsic weights of the model are automatically changed numerous times during the training stage. This training process is influenced by external variables such as training strategy, architecture, and regularisation approaches.

Comparing research and its results is difficult since they do not utilise the same data and do not provide all of the necessary features to replicate their experiments. The impacts of ordering are also examined several times in these studies. However, we choose to compare training and architectural techniques while keeping in mind that some of their outcomes may be incomplete.

3.4 Training Strategies:

There are two methods for CNN training: scratch and transmission. Transmission learning is enabled via a network of pre-trained pictures (e.g. ImageNet and its 1.2 million photos in 1000 classes). The fact that the first layers of CNNs are generically low-level characteristics distinct from classes allows for this type of learning. To modify this in practice, the network weights from prior sessions are used. When just a little amount of training data is available, such as in the case of agricultural disease diagnosis, transfer learning allows us to employ CNNs.The selection of a training course is based on both technical (number of pictures, processing capacity) and topical considerations (accessibility of an acceptable architecture or a weight of pretrained compatible with the data utilised). Brahimi (2018) examined three CNN architectural approaches (AlexNet, DenseNet-169, Inception v3, ResNet-34, SqueezeNet-1.1, and VGG13). They utilised a background lecture on the data set PlantVillage.Feature extraction and optimum correction were both employed. The network was trained from the ground up using the third method.

The six designs were fine-tuned for optimum accuracy (from 99.2 percent for SqueezeNet to 99.5 percent for VGG13). For proper tune and training, close times are required (from 1.05 to 5.64 h for fine-tuning and from 1.05 to 5.91 h when trained from scratch). The extraction process requires the least amount of training (from 0.85 to 3.63 h). Scratches and transfer learning should not be viewed as mutually exclusive techniques. However, if a new architecture is not available for past weights, pre-training may be beneficial.

It seeks to start from the ground up with the creation and finalization of a model based on big data sets (for example, ImageNet and PlantVillage or other database).

4. MODEL ARCHITECTURE

Convergence layers, pooling layers, and active functions are three key components of CNN (ReLUs). The number of layers utilized, how they're organized, and whether or not additional treatment units are created differ from one design to the next. The most sophisticated architecture was chosen in 17 of the 19 situations examined.

In 11 studies, the designs described in conjunction with the first popular CNNs are employed. ImageNet claims to have conducted research using SqueezeNet, which has 50 times less parameters than AlexNet. Fifteen research projects make use of the well-known ResNet, VGG, and Inception architectures.. Four of these studies were compared with alternative designs for maximum and lowest accuracy in 12 of 19 tests. The accuracy of ResNet-101 varies from 59 percent to 99.75 percent for architecture in DenseNet-121, and (Too). The advantage of VGG-16 over ResNet-101 findings for Fuentes demonstrates that network complexity and depth do not make up to accuracy (2017). Abraham (Abraham) (Brahimi). From research to research, the consequences of a unique design might change, like with the VGG-16, which was deemed the best of Wang's archaeology. Different architectures are being evaluated to determine the best architecture for a given situation, since architectural deployment is becoming more common in standard libraries like as PyTorch and Tensorflow/Keras. The kind, amount, time, and resources of the material supplied for training will be determined by this choice. The architectural selection and calculation of optimum values for distinct hyperparameters may be a dangerous test and error technique. This process, however, is regulated by techniques, whether manual tweaking, random sampling, or grid research (Bengio).

A total of seven research made use of customized architectures. To manage the research data, a benchmark architecture was required. Oppenheim uses a standardization approach based on the random connection between model layers to reduce the danger of overfitting owing to their tiny data locations. Architectures were more personalized in some situations. Cruz used a technique known as "Abstraction Level Fusion." They may speed up in the completely connected stages of the guide by manually inserting functions. To make the best use of colour information, Zhang built a three-channel neural network.

4.1 Regularization Techniques

The most difficult aspect of machine learning is creating a trained model that can analyse fresh and previously unknown data. There is no guarantee that the instruction will be accurate to a high degree. Deep learning is, in fact, the worst blunder. This occurs when the number of input samples is insufficient in comparison to the network's learning capacity. Overfitting prevents learning of the classes' primary features and instead captures the training set's noise (Srivastava) This results in an incredibly accurate model that cannot be generalized during training (i.e., it does not achieve high test accuracy). Separate data is rarely used to test trained corpus models (only two studies have an explicitly independent test set). As a result, it is impossible to predict when the models will be overhauled. However, a small minimum number of samples was given in many of the research chosen per class (before increment): equal or below 55 in two studies and between 55 and 200 in eight investigations. In certain situations, the number of images is insufficient to train a model reliably, particularly given the diversity of the plant world.

The simplest and most apparent solution is to acquire additional data to improve model generalization. However, some images are difficult to get in an agricultural setting, particularly when dealing with illnesses. Machine learning has improved the performance of the test set while decreasing the performance of the training set. Regularization procedures are referred to as. To begin with, an increase in data relating to the shape or intensity of fresh pictures. Rotation, reflection, random cuts, zooms, even noise-lessness may be added, contrast levels can be changed, and new situations can be simulated. Overall, the results are straightforward. As a result, the data set's size and diversity are artificially enhanced. The increase can be done in a variety of ways, with each picture including one or more modifications (perhaps chosen at random). Changes can also be adjusted "online" before the whole exercise begins or for each frame batch. Some investigations should highlight a flaw: the number of images rose before they were divided into training sets. When the sets are configured so that a picture and its copies are in the same set, these increase actions must be done.

Additional strategies, such as weight loss and precocious cessation, have been found in the studies, such as discontinuance or formation. These methods are recommended.

5. SYSTEM ANALYSIS

Informal scientists have been attempting to construct computers that feel like visual data since the 1950s, when artificial intelligence was in its infancy. The field of computer vision made minimal development during the next few decades. In 2012, a group of academics at the University of

Toronto developed a model for IA that outperformed the biggest image recognition systems.

The AI system AlexNet (named after its developer, Alex Krizhevsky) took first place in the ImageNet 2012 Computer Vision Competition with an impressive 85 percent accuracy. The rider gave the test a moderate rating of 74%.

The AlexNet was built on a specialized type of artificial neural network that closely resembles the human visual system. In recent years, a large number of CNNs have shown to be critical for a variety of computer vision applications. This is what CNNs should know about their origins and operations

5.1. The Brain

We examine the world around us on a regular basis. We anticipate and respond to what we see without even realizing it. When we see something, we may recognize it based on what we've learned in the past. Consider the photo in figure 2 for a moment

Figure 2: Early analysis of a scene

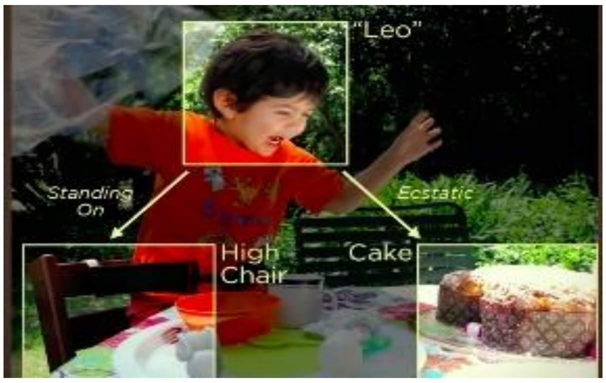


Figure 3: An observation of the Brain

You could have thought, "That's a cheerful little child on a chair," as seen in Figure 3. Probably, you were pondering something. Or you probably mistook his wailing for an attack on the dessert in front of him. Throughout the day, we do so unconsciously. We keep track of classification, predicting, and recognition trends. But what are our options? How can we make sense of what we see as in Figure 4? It took nature 500 million years to create a mechanism to do this. The primary visual circuits, which include the eyes and brains, collaborate to make the world around us meaningful.

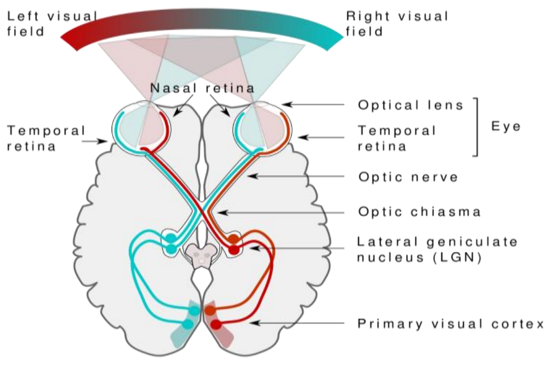


Figure 5: Response to visual stimuli in a Brain

What we experience is truly comprehended in the brain, the main visual cortical cortex, as soon as our eyes begin to gaze. Figure 5 illustrates this. When you look at anything, the light receptors in your eyes send signals to the primary visual brain through the optic nerve, which is where information is processed. The eye is sensed by the primary visual brain.

5.2 Convolutional Neural Networks

To train an image algorithm, we utilize a special type of Artificial Neural Network. Its name is derived from one of the network's most essential functions: convergence. CNN was influenced by the human brain. In Hubel's mammalian brain studies in the 1950s and 1960s, a novel model for visual perception of the mammalian environment was presented. They demonstrated that cats' and monkeys' visual cortexes are neurons that respond to neurons in their immediate surroundings.

In their investigation, they identified two main kinds of neuronal visionary cells in the brain that behave differently (C cells.

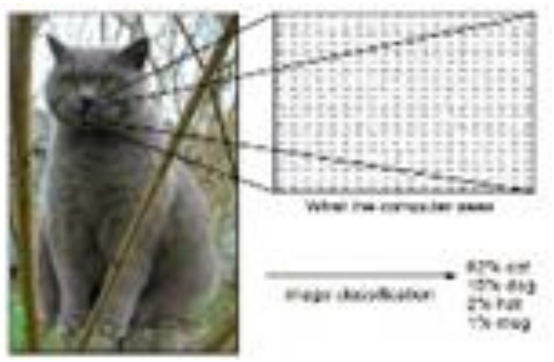


Figure 4: How a computer sees an image

Fukushima, a Japanese scientist, proposed a brain system hierarchy in 1980. He dialled the number for the recognition. The concept of simple and complex cells was used to create this model. By learning about object shapes, the recognition was able to recognize patterns.

Bengio's research on Convolutionary Neural Networks was first published in 1998. The first Convolutionary Network was LeNet-5, and human input numbers may be classified.

6. RESULTS AND OBSERVATIONS

6.1. Implemented Code

Load the libraries and import the dataset

To train our model, we import all of the necessary components. The import modules used for training include:

- **ImageDataGenerator**
  Image Data Generator generates image data tensors in batches in real-time. The output photographs of the generator are identical to the input photos. Increased picture data is a technique for artificially increasing the data size by inserting updated image copies into the data set. With more data, deep model training for neural networks can result in better models. The Image Data Generator class may be customized using the Kera neural network's deep learning library.

- **Conv2D**
  This layer creates a kernel that converts a layer input to a tensor output. The cooling matrix or mask in the image processing kernel is used to blur a kernel or picture by sharpening, taping, edge detection, and other methods.

- **MaxPooling2D**
  Procedure for pooling data in 2D space to the maximum extent possible. Procedure Download the input value across the pond size window by taking the maximum value for each dimension axis. The window may be customised with steps in each dimension.

- **Dropout**
  The drop-off layer affects every stage of exercise with an overfit frequency. The non-0 input is electrically enhanced by 1/(1), resulting in a total that remains constant across all inputs. The drop-off layer only applies when the practice is set to True, thus no values are discarded. If you use a model. Fit the training is automatic, and if the layer is used in other circumstances, you can explicitly set kWarg to true. In our training model, we used the drop-off value of 0.25.

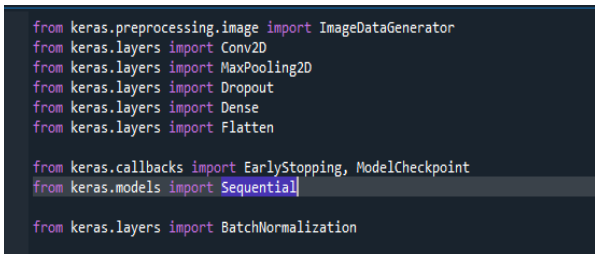
- **Dense**
  The layer of a typical neural network is dense. It's the one that's most frequently used.

- **Flatten**
  The entrance is free-flowing. Do not alter the lot's dimensions. Between the convolutional layer and the fully connected layer comes a flattening layer. A vector is created by a linked neural network classifier. It is possible to categorise neural networks in an integrated manner. Requests.

- **Early Stopping**
  The amount of time that can be spent training neural networks is a constraint. The training data set may be over-adjusted too frequently, while a few may result in a poor model. Early stopping is a strategy for setting multiple training intervals and ending the training when the model's performance has been achieved and the data package for validation has been completed. In our train model, we used patience=10 (Number of epochs with no improvement after which training will be stopped).

- **Sequential**
  The Keras Python package makes building deep learning models simple and quick. The sequential API can construct a layer-by-layer model in the vast majority of situations. It's restricted to prohibit the construction of models with many inputs or outputs or for layer sharing.

- **Batch Normalization**
  A deep neural learning network's load standardization refers to the standardization of entries on a single layer. The batch standardization technique substantially accelerates the neural network training process while also improving the model by introducing a modest regulatory effect in specific circumstances. The data is transformed into standard inputs, which include a zero and standard deviation. Figure 7 illustrates the code snippets.



```
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dropout
from keras.layers import Dense
from keras.layers import Flatten

from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.models import Sequential

from keras.layers import BatchNormalization
```

Figure 4: Code snippets of import modules

6.2. Data Preprocessing

When image data is pre-processed, it is transformed into useful floating-point tensors that are fed into neural networks. Tensors are multidimensional arrays used to store data. A tensor's dimensions with a 64x64 picture and three channels (64, 64, 3). The data is now recorded on a disc as JPEG files, allowing the procedures to be followed.

6.3 Model Creation

The CNN model is completely integrated and consists mostly of concentration layers, concentration, and decomposition. Because it performed better with grid data, CNN was able to solve the photo categorization challenge effectively. During the exercise, the drop layer may deactivate neural components, lowering the model's fitness. The Adam Optimizer has been created. We use the "add()" function to add layers to our model..

The first two levels are Conv2D. Condensed layers in two-dimensional matrices are used as input images. There are sixty-four nodes in the first and second levels. The amount might be modified to more or less depending on the scope of the data gathering. We had a lot of success with 64 and 32, so we'll stick with that for the time being. Our filter matrix is the size of the kernel. As a result, a 3x3 matrix filter is proposed with a kernel size of 3.. The layer function of activating the layer is a layer function. For the first two layers, we utilize the Rectified Linear Activation (ReLU). In neural networks, this function performs well. The input form is also a part of our first layer. This is the format of each image you upload.

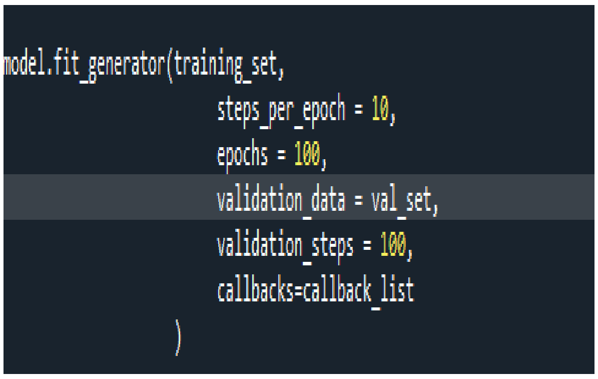
6.4 Compiling the model

Following that, we must construct our model. Model creation necessitates the use of three components: an optimizer, a loss, and measurements. The optimizer controls the learning rate. We'll use 'adam' as our optimizer. Adam is an excellent optimizer in a variety of situations. The adam optimizer changes the learning rate during training.

6.5 Fitting the Model

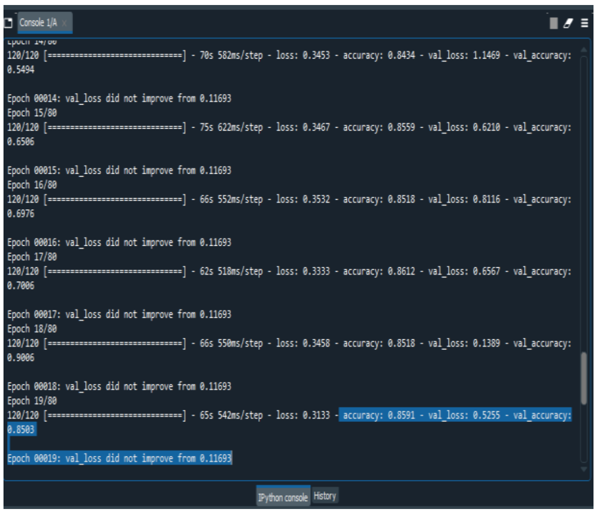
To fit the model into data, use the fit generator technology, which is the same as the data generator. His first objective is to develop a Python generator that can generate an infinite number of inputs and targets since data is always being generated. The Keras model must know how many samples the generator may draw before declaring an era. Each epoch parameter's steps will be accountable.

6.6 Code



```
model.fit_generator(training_set,
                    steps_per_epoch = 10,
                    epochs = 100,
                    validation_data = val_set,
                    validation_steps = 100,
                    callbacks=callback_list
                )
```

Figure 8: Code



```
120/120 [==============================] - 70s 582ms/step - loss: 0.3453 - accuracy: 0.8434 - val_loss: 1.1469 - val_accuracy:
0.5494

Epoch 00014: val_loss did not improve from 0.11693
Epoch 15/80
120/120 [==============================] - 75s 622ms/step - loss: 0.3467 - accuracy: 0.8559 - val_loss: 0.6210 - val_accuracy:
0.6506

Epoch 00015: val_loss did not improve from 0.11693
Epoch 16/80
120/120 [==============================] - 66s 552ms/step - loss: 0.3532 - accuracy: 0.8518 - val_loss: 0.8116 - val_accuracy:
0.6976

Epoch 00016: val_loss did not improve from 0.11693
Epoch 17/80
120/120 [==============================] - 62s 518ms/step - loss: 0.3333 - accuracy: 0.8612 - val_loss: 0.6567 - val_accuracy:
0.7006

Epoch 00017: val_loss did not improve from 0.11693
Epoch 18/80
120/120 [==============================] - 66s 550ms/step - loss: 0.3458 - accuracy: 0.8518 - val_loss: 0.1389 - val_accuracy:
0.9006

Epoch 00018: val_loss did not improve from 0.11693
Epoch 19/80
120/120 [==============================] - 65s 542ms/step - loss: 0.3133 - accuracy: 0.8591 - val_loss: 0.5255 - val_accuracy:
0.8503

Epoch 00019: val_loss did not improve from 0.11693
```

Figure 9: code

6.7 Model Training and Evaluation

Following Models were trained

1. Model 1
- Batch size =8
- Steps per epoch =100
- Epochs = 100
- Validation steps =25

Resulted Model
- Loss = 0.1528
- Accuracy = 0.9271
- Validation Loss = 0.029
- Validation Accuracy = 0.9822

Actual accuracy: 80%

2. Model 2
- Batch size =8
- Steps per epoch = 120
- Epochs = 236
- Validation steps =32

Resulted Model
- Loss = 0.1373
- Accuracy = 0.9321
- Validation Loss = 0.0941
- Validation Accuracy = 0.9924

Actual accuracy: 82

3. Model 3
- Batch size =16
- Steps per epoch =126
- Epochs = 230
- Validation steps =200

Resulted Model
- Loss = 1.4922
- Accuracy = 0.2527
- Validation Loss = 1.6881
- Validation Accuracy = 0.0018

Actual accuracy: 60%

4. Model 4
- Batch size =16
- Steps per epoch =290
- Epochs = 270
- Validation steps =280

Resulted Model
- Loss = 1.4816
- Accuracy = 0.1398
- Validation Loss = 1.8217
- Validation Accuracy = 0.1299

Actual accuracy: 90%

5. Model 5
- Batch size =32
- Steps per epoch =30
- Epochs = 196
- Validation steps =40

Resulted Model
- Loss =1.3230
- Accuracy = 0.0426
- Validation Loss = 1.8822
- Validation Accuracy = 0.1184

Actual accuracy: 87%

6. Model 6
- Batch size =16
- Steps per epoch =60
- Epochs = 4
- Validation steps =100

Resulted Model
- Loss = 1.7923
- Accuracy = 0.1708
- Validation Loss = 1.7975
- Validation Accuracy = 0.1675

Actual accuracy: 89%

7. Model 7
- Batch size =32
- Steps per epoch =70
- Epochs = 8
- Validation steps =100

Resulted Model
- Loss = 1.4659
- Accuracy = 0.1696
- Validation Loss = 1.7930
- Validation Accuracy = 0.9257

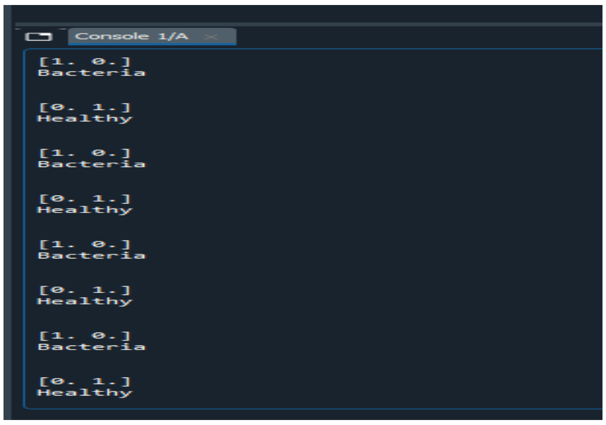
Actual accuracy: 80%

8. Model 8
- Batch size =64
- Steps per epoch =70
- Epochs = 100
- Validation steps =120

Resulted Model
- Loss = 1.6333

- Accuracy = 0.6729
- Validation Loss =0.0062
- Validation Accuracy = 0.9999

Actual accuracy: 95%



```
[1. 0.]
Bacteria

[0. 1.]
Healthy

[1. 0.]
Bacteria

[0. 1.]
Healthy

[1. 0.]
Bacteria

[0. 1.]
Healthy

[1. 0.]
Bacteria

[0. 1.]
Healthy
```

Figure 10: Final Output

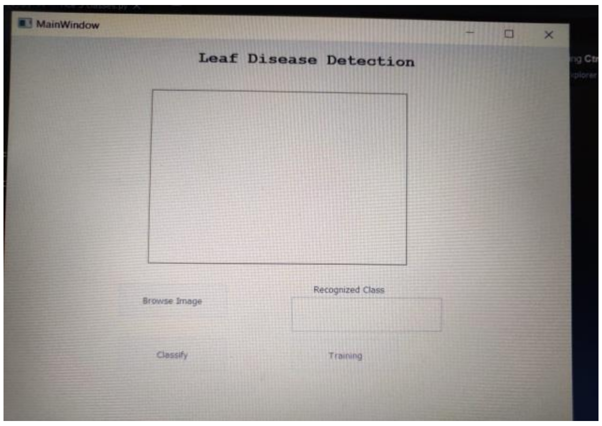


Figure 11: Graphical User Interface

A Graphical User Interface is created with buttons for three purposes:

- Training the model.
- Browsing the image for testing.

Classifying the image

After training the model, we click on the button to browse an image as shown in figure 12.

We select an image from the list of images as depicted in image 13

After an image is selected, it is displayed in the GUI and then we move further to classify it. As depicted in image 14

On clicking the "Classify" button, the disease of the image is displayed as one of the five categories of leaves as depicted in Figure 15

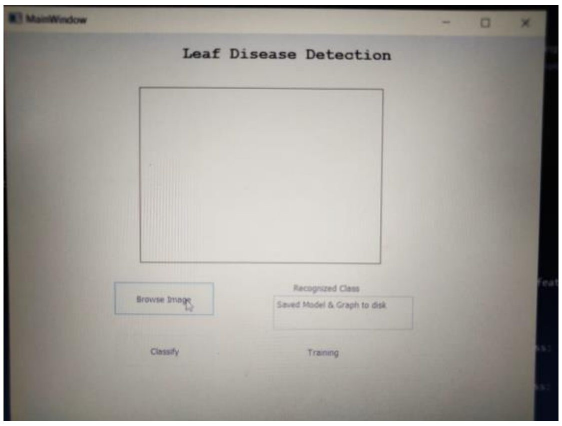


Figure 12: Browsing an image after training model

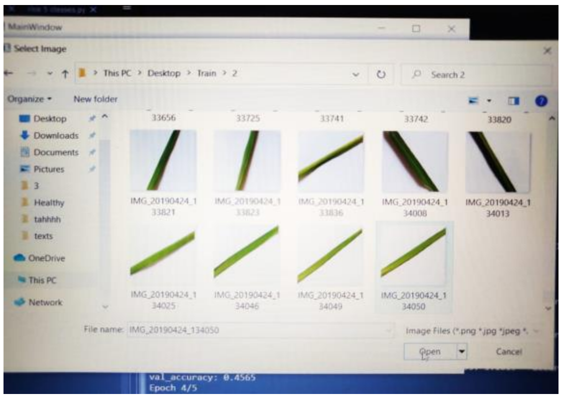


Figure 13: Selection of an image

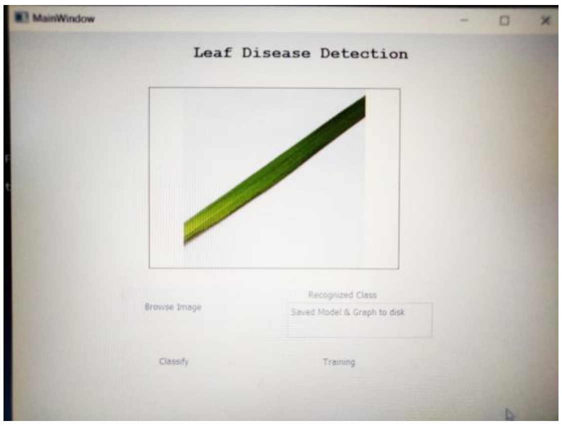


Figure 14: Display of image in GUI

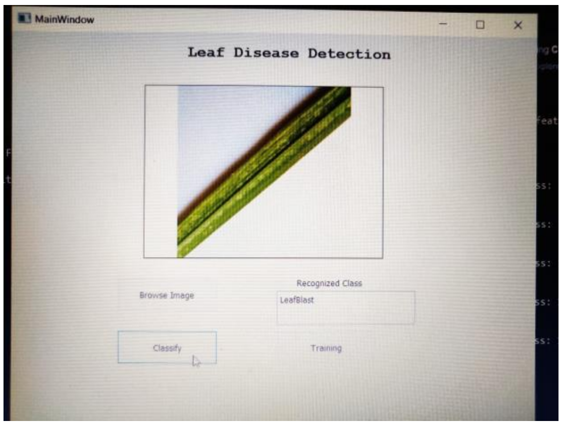


Figure 15: The classify button

Disease detection: Some of the results that we got can be seen from Figure 16 to Figure 22. In these results are mentioned if the leaf is healthy or diseased, and if the leaves are diseased, the type of disease is also shown/ mentioned as results.

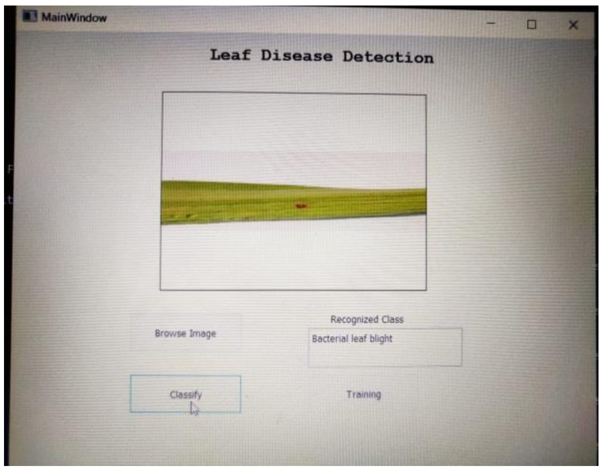


Figure 16: Bacterial leaf blight

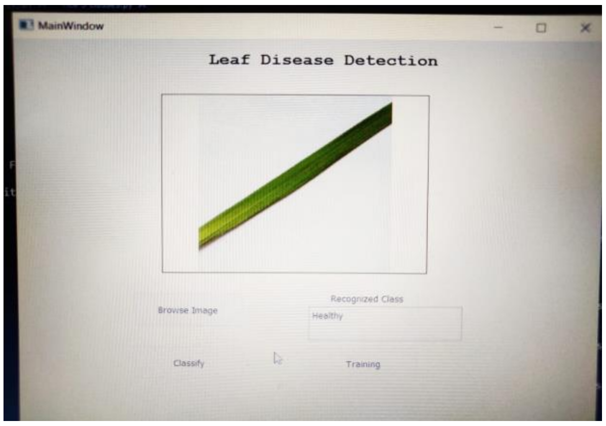


Figure 17: Healthy

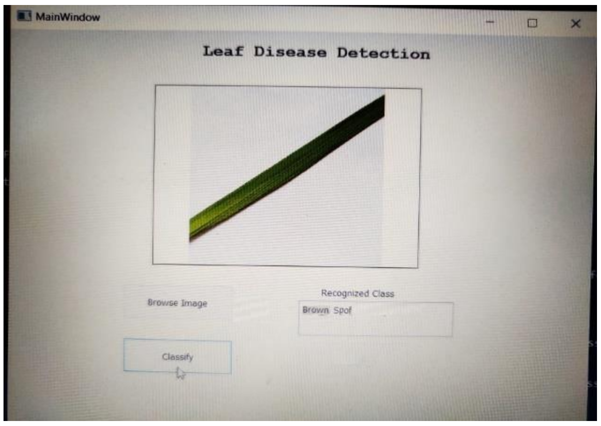


Figure 18: Brown Spot

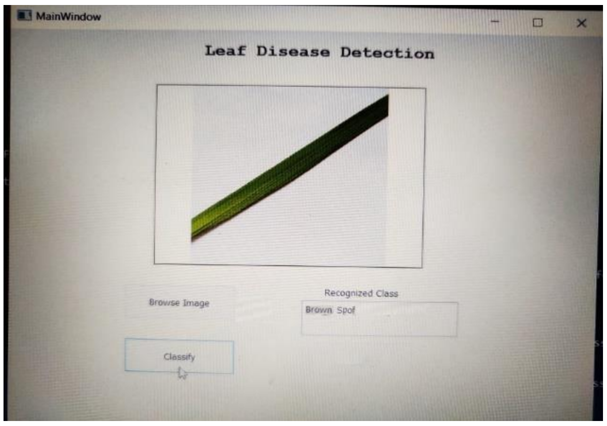


Figure 19: Leaf smut

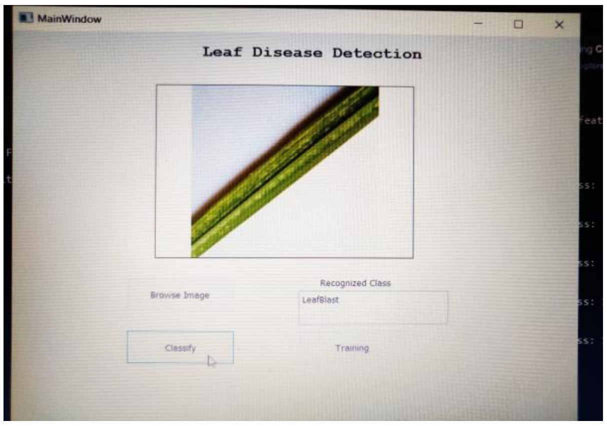


Figure 20: Leaf Blast

7. CONCLUSION

It is critical to have an accurate diagnosis and categorization of rice varieties in order to avoid crop loss. Based on image processing techniques, the suggested model successfully categorised and identified rice leaf illnesses. This model is built using the CNN machine learning method. The implementation uses a graphical user interface (GUI). The suggested approach uses healthy and disease-affected rice plant leaves to distinguish healthy and unhealthy features. Following that, the suggested model is used to analyse the pictures and classify the leaf as either diseased or healthy. To categorise our findings, we used five different categories. This proposed model has a 95 percent accuracy rate. This model correctly distinguishes between infected and healthy rice plants.

ACKNOWLEDGEMENT

I am extremely thankful and obliged to my guide Dr.Jasmeen Gill and HOD of CSE Department. RIMT University for guiding me during the course of my research. I am thankful to my parents for supporting me morally and financially.

REFERENCES

1) Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 3642–3649. IEEE (2012)
2) Cires¸an, D.C., Giusti, A., Gambardella, L.M., Schmidhuber, J.: Mitosis detection in breast cancer histology images with deep neural networks. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2013, pp. 411–418. Springer (2013)
3) Ciresan, D.C., Meier, U., Masci, J., Maria Gambardella, L., Schmidhuber, J.: Flexible, high performance convolutional neural networks for image classification. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence. vol. 22, p. 1237 (2011) Introduction to Convolutional Neural Networks 11
4) Cires¸an, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Convolutional neural network committees for handwritten character classification. In: Document Analysis and Recognition (ICDAR), 2011 International Conference on. pp. 1135–1139. IEEE (2011)
5) Egmont-Petersen, M., de Ridder, D., Handels, H.: Image processing with neural net- worksa review. Pattern recognition 35(10), 2279–2301 (2002)
6) Farabet, C., Martini, B., Akselrod, P., Talay, S., LeCun, Y., Culurciello, E.: Hardware accelerated convolutional neural networks for synthetic vision systems. In: Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on. pp. 257–260. IEEE (2010)
7) Hinton, G.: A practical guide to training restricted boltzmann machines. Momentum 9(1), 926 (2010)
8) Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Im- proving neural networks by preventing co-adaptation of feature detectors. arXiv
9) preprint arXiv:1207.0580 (2012)
10) Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on 35(1), 221–231 (2013)
11) Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large- scale video classification with convolutional neural networks. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. pp. 1725–1732. IEEE (2014)
12) Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convo- lutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
13) LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation 1(4), 541–551 (1989)
14) LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
15) Nebauer, C.: Evaluation of convolutional neural networks for visual recognition.
16) Neural Networks, IEEE Transactions on 9(4), 685–696 (1998)
17) Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: null. p. 958. IEEE (2003)
18) Srivastava, N.: Improving neural networks with dropout. Ph.D. thesis, University of Toronto (2013)
19) Szarvas, M., Yoshizawa, A., Yamamoto, M., Ogata, J.: Pedestrian detection with convolutional neural networks. In: Intelligent Vehicles Symposium, 2005. Proceedings. IEEE. pp. 224–229. IEEE (2005)
20) Szegedy, C., Toshev, A., Erhan, D.: Deep neural networks for object detection. In: Advances in Neural Information Processing Systems. pp. 2553–2561 (2013)
21) Tivive, F.H.C., Bouzerdoum, A.: A new class of convolutional neural networks (siconnets) and their application of face detection. In: Neural Networks, 2003
22) V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In Proc. 27th International Conference on Machine Learning, 2010.
23) N. Pinto, D.D. Cox, and J.J. DiCarlo. Why is real-world visual object recognition hard? PLoS computational biology, 4(1):e27, 2008.
24) N. Pinto, D. Doukhan, J.J. DiCarlo, and D.D. Cox. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. PLoS computational biology, 5(11):e1000579, 2009.
25) B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. Labelme: a database and web-based tool for image annotation. International journal of computer vision, 77(1):157–173, 2008.
26) J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pages 1665–1672. IEEE, 2011.