PADDY LEAF DISEASE DETECTION USING CNN-LSTM MODEL

Anuj Gaida
Department of Computer Engineering
Khwopa College of Engieering,TU
Bhaktapur, Nepal
gaidaanuj@gmail.com

Bikesh Manandhar
Department of Computer Engineering
Khwopa College of Engieering,TU
Bhaktapur, Nepal
bikeshmanandhar9@gmail.com

Er. Dinesh Ghemosu
Department of Electrical Engineering
Khwopa College of Engieering,TU
Bhaktapur, Nepal
ghemosu.dinesh@gmail.com

Rahul Khatri
Department of Computer Engineering
Khwopa College of Engieering,TU
Bhaktapur, Nepal
rahulkc1290@gmail.com

Safal Raj Manandhar
Department of Computer Engineering
Khwopa College of Engieering,TU
Bhaktapur, Nepal
Safalm74@gmail.com

***Abstract*—In recent years, automated image classification techniques have gained prominence across various sectors like agriculture, medicine, e-commerce, and facial recognition. In agriculture, these techniques are used for tasks such as identifying weeds, categorizing fruits, and detecting plant diseases. They enable quicker, more accurate decision-making and contribute to efficiency, sustainability, and improved resource management in the agricultural sector. In this article, we focused on classifying whether the inputted image is healthy or has symptoms of three types of commonly observed paddy diseases namely, brown spot, leaf blast, and hispa. Firstly, the diseased area is segmented out, which is then forwarded to the neural network for classification. Our proposed model is a combination of CNNs and LSTM, used to classify the type of disease by utilizing the segmented image. In this article, the combination of a single CNN model with LSTM and the combination of multiple CNN models with LSTM are performed, and we observed that when combining two CNN models, accuracy is much higher than combining a single CNN model to LSTM model due to increase in the number of features for classification. As simply increasing the number of sequential data does not necessarily mean an increase in accuracy. So, when testing with different numbers of pixel data in a single sequence we observed to have the highest test accuracy of 89.67%. By optimizing various hyperparameters we obtained the highest test accuracy of 95.50%. During hyperparameters optimization, we also observed that having a large number of real-time data augmentation also can have a huge impact on model accuracy.**

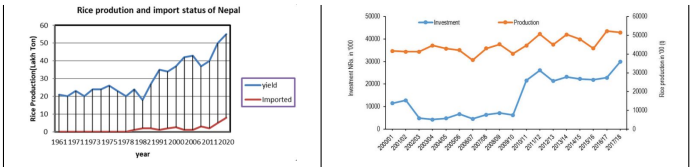
***Index Terms*—Convolution Neural Network, Image Segmentation, Long Short Term Memory**

I. INTRODUCTION

RICE is a vital staple crop worldwide, particularly in Asian countries like Nepal. However, rice plants are susceptible to various diseases that can cause significant damage if not detected early. Manual classification of these diseases is labor-intensive, expensive, and time-consuming, making it crucial to develop automated systems for disease identification. Machine learning and deep learning techniques have shown promise in automating disease detection in trees and plants, including rice. This paper presents a novel approach using deep learning methods to detect and classify three common rice diseases: Brownspot, Hispa, and Leafblast.

In accordance to the survey report of "Ministry of Agriculture and Livestock Development". Nearly 70% of Nepal's population depends directly on agriculture, and contributes more than a quarter of the country's GDP – with rice making up 21% of that. Till as late as 1985, Nepal used to be a net exporter of rice, and during the 1960s the country was exporting up to $45 million worth of rice to India every year. How the tables have turned, in 2015 Nepal has to import 531,000 tons of rice worth $210 million from India.

The reference graph is taken from [Opportunities and Challenges in Irrigation Practices and Agricultural productivity scenario in Nepal: A Review] and [Dynamics of rice subsector in Nepal: Research investment, production, and supply chain] which illustrate the Nepal Rise Production in different fiscial year.



Rice production & Import | Rice Production & Investment

Fig. 1. Survey Detail of Rice Production

Based on the [Rice Science and Technology in Nepal], here are list of the paddy diseases that were observed in Nepal:

S.N	Name of Disease	Distribution	Year	Status
1	Seed-bed beetle	Kathmandu	1962	Minor
		Sundarijal, Nayapati, Sunkhani, Dharmasthali	1966	Major
2	Mealybug	Janakpur	1964	Minor
		Sarlahi, Bara, Parsa, Rautahat, Dhanusha	1966, 1967	
		some areas of Terai rice belt	1968	Major
		Parwanipur	1980	Major
3	Stem gall midge	Entire Terai	1971	Minor
4	Leaf folder	Dhanusa, Jaleswor, Sarlahi, Bara, Parsa	1977	Major
		Kathmandu, Bara, Godavari, Bandegaon, Janakpur, Parsa	1978	Major
		Parwanipur, Janakpur, Kankai, Bara, Parsa	1980	Major
5	Hispa	Kathmandu	1962	
		Terai belt, Sunsari, Morang	1970	Major
		Bara, Sarlahi, Dhanusha, Mahoari, Jhapa	1971	Major
		Bara, Parsa	1972	
		Parsa	1973	
		Bara, Hectares, Rautahat	1975	
		Lalitput	1982	
6	Stem Borers	Bhaktapur	1960	Major
		Kathmandu	1965, 1967, 1970, 1982	
		Parwanipur	1969	
		Bardia		
8	Armyworm	Trisuli	1973	
		Kailali, Kanchanpuin	1974	Major
		Chitwan, Dang, Baglung	1977	Major
		Chitwan	2013	
9	Whitefly	Chitwan, Tanahun, Lamjung, Gorkha, Kaski	2005	Major
10	Hopper	Kathmandu	1962, 1964, 1965, 1968	
		Biratnagar	1968	Minor
		Chandradangi, Jhapa	1977, 1978	Major
		Parwanipur, Khumaltar	1962	
		Kathmandu	1982	Major
		Kumroj, Kathar, Chitwan	1996	Major
		Surkhet	2004	Major
11	Seed bug	Kathmandu	1962	Major
		Parwanipur	1963	
		Janakpur	1972	
		Bhairahawa	1996	

TABLE I
RECORDED PADDY DISEASES

From the recorded data of various rice diseases in Nepal and the provided graph, We can observe Rice production in Nepal declined significantly from 1965 to 2000 due to diseases, with varying degrees of reduction in rice production from 10-20% to above 80% in different areas. Despite some improvement in the past decade, it remains insufficient to meet the needs of the growing population. Increased production costs have negatively impacted Nepalese farmers, primarily due to the excessive and unmanaged use of disease prevention medicines without proper identification. This misuse harms both the plants and the soil, rather than enhancing land productivity.

Numerous AI models have been employed for the diagnosis of plant diseases and have been successful in applications including plant identification and disease detection. Despite this, these models have the drawback of not being able to extract several important visual attributes from the input image [1]. However, the accuracy of these models has been an interest of the field to many researchers. CNN models alone are unable to calculate the dependency and continuity features of the intermediate layer output [1]. Therefore, the combination of CNN with the RNN model can help to improve the accuracy. As recurrent neural network can be utilized to link the middle tire features to the final fully-connected network for classification [2].

In summary, this paper has the following contributions:

- We introduced a new approach of combining CNNs models and LSTM layer.
- We utilized multiple CNNs models for feature extraction aimed at maximizing the utility and effectiveness of the limited data available to improve model performance and LSTM layer was utilized to capture temporal dependencies between the extracted feature before final prediction.

In the following sections, we provide a comprehensive overview of our research findings on (PADDY LEAF DISEASE DETECTION USING CNN-LSTM MODEL). We begin with a discussion on the existing literature in section II (LITERATURE SURVEY), identifying gaps in the current research landscape. Subsequently, we present our proposed methodology in section III (METHODOLOGY). In section IV (RESULT AND DISCUSSION), we describe the evaluation metrics used and provide a comparison of our model with other approaches. Finally, we conclude with a summary of our findings and discuss avenues for future research in section V (CONCLUSION AND FUTURE ENHANCEMENT).

II. LITERATURE SURVEY

In 2022, Garg et al. [1], This paper suggests combining a specific kind of RNN called LSTM with a pre-trained CNN network. Transfer learning was used to separate the deep features from the Xception and VGG16 pre-trained deep models' several fully connected layers. To help the proposed model be more focused the recovered deep features from the CNN layer and RNN layer were concatenated and sent into the fully connected layer.

In 2018 by Islam et al. [3] a model was created to categorize the disease based solely on the extracted percentage of the RGB value of the damaged area of rice leaf using image processing. To finally classify the diseases into three disease classes Bacterial leaf blight, Rice blast, and Brown spot the RGB percentages were fed into a Naive Bayes classifier. The model's disease classification accuracy is over 89%.

Islam et al. at 2020. [4] proposed a Combined deep CNN-LSTM network, where 4575 images of pneumonia and normal cases were used. The network of 20 layers: 12

convolutional layers, 5 pooling layers, one fully connected layer, and one lstm layer with an input size of 224*224 pixels was used to obtain 99.4% accuracy with an AUC score of 99.9%.

Another paper by Rangarajan et al. at 2018. [5] used the pre-trained DL and its algorithm to accomplish the categorization of disease in tomato plants. The authors used AlexNet and VGG16 pre-trained architectures on PlantVillage consisting of 13262 pictures, both of which obtained 97.49% classification accuracy.

Karthik et al. at 2020. [6],proposed embedded residual CNN where two distinct deep architectures are shown for identifying the type of infection in tomato leaves. Relative learning is used in the first architecture to identify important elements for categorization. The attention mechanism is added to the remaining deep network in the second architecture. The suggested work used the attention mechanism to take advantage of the characteristics that the CNN learnt at different levels of the processing hierarchy, and obtain overall accuracy of 98% on the validation sets in the 5-fold cross-validation.

Another paper by Lamba et al. at 2022 [7] proposed a brand-new hybrid model based on neural networks (GCL). Long-short-term memory (LSTM) and convolutional neural networks are combined in GCL to supplement datasets. The dataset is enhanced using GAN, CNN extracts the characteristics, and LSTM categorizes the various paddy diseases. To increase the precision and dependability of the classification model, the GCL model is being examined.

In 2021 Krishnamoorthy et al. [8] utilized InceptionResNetV2 a type of CNN model with transfer learning approach for recognizing diseases in rice leaf images. The parameters of the proposed model is optimized for the classification task and obtained a good accuracy of 95.67%.

Al-Amin et al. at 2019, December [9] proposed a CNN based model that provides 97.40% accuracy in predicting various diseases of rice leaves. Using a data set of over 900 images of diseased and healthy leaves and following the technique of 10-fold cross validation(CV), the model was trained to identify 4 common rice diseases.

III. METHODOLOGY

The primary objective of this project is to develop a deep learning-based model for accurate detection of rice leaf diseases, aiming to achieve a high level of precision in recognizing and classifying the diseases. The data for this task is collected from kaggle which is a popular online platform that provides access to a wide range of dataset for AI model training. And, all these models are trained on Google Colab, a cloud-based platform provided by Google that offers a convenient and powerful environment for training AI models.

A. Dataset Description

The main dataset used for disease classification was downloaded from Kaggle[1]. This dataset was originally 9.12 GB containing high-quality images of 3 different diseases and healthy classes. The size of the images is not constant throughout the dataset(min size = 1200*1200, max size = 2163*2163). The initial dataset was augmented to achieve uniform pixel dimensions of 256x256, filter out blurry images, and expand the dataset. The description of the resulting dataset is presented below.

Brown Spot	523
Leaf Blast	799
Hispa	565
Healthy	1488

TABLE II
DATASET DESCRIPTION

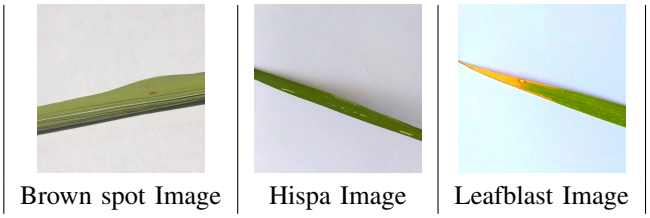


Fig. 2. Sample images

B. System Workflow

1) Image Segmentation: The initial step in the process involves resizing the images in the dataset to a resolution of 256x256 pixels using the relevant function provided by the OpenCV library. Subsequently, a combination of thresholding methods, specifically binary and Otsu thresholding, is applied to the images. Furthermore, morphological operations are performed on the output of the Canny edge detection to accurately segment and isolate the diseased parts of the images.

2) Model Training: The initial step involves importing the necessary Python libraries, including Numpy, OpenCV, Matplotlib, and TensorFlow. Subsequently, the model architecture is defined, which comprises two pre-trained CNN models (VGG16 and VGG19) trained on the local dataset. These models are combined with an LSTM layer and a fully connected dense layer consisting of four nodes. Following the model architecture definition, various compile parameters are set. Among these parameters, two important ones are the optimizer and the loss function.

[1]https://www.kaggle.com/datasets/shayanriyaz/riceleafs

The optimizer used in our model is Adam, which stands for Adaptive Moment Estimation. Adam is an optimization algorithm commonly used in deep learning. It combines the benefits of two other popular optimization algorithms, namely AdaGrad and RMSProp. Adam dynamically adjusts the learning rate based on the gradients of the parameters, allowing for efficient and effective training of neural networks.

Here is the mathematical expression for the parameter update step in Adam:

1 Initialize time step t = 0.
2 Initialize the first and second moment variables:
   - First moment estimate: m = 0 (initialized as a vector of zeros, with the same shape as the parameters)
   - Second moment estimate: v = 0 (initialized as a vector of zeros, with the same shape as the parameters)
3 At each time step t:
   - Compute the gradient of the loss function with respect to the parameters:

$$g_t = \nabla_\theta L(\theta_{t-1}) \tag{1}$$

   - Update the first moment estimate:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{2}$$

   - Update the second moment estimate:

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{3}$$

   - Bias-corrected first moment estimate:

$$m_{t_{\text{hat}}} = \frac{m_t}{1-\beta_1^t} \tag{4}$$

   - Bias-corrected second moment estimate:

$$v_{t_{\text{hat}}} = \frac{v_t}{1-\beta_2^t} \tag{5}$$

   - Update the parameters:

$$\theta_t = \theta_{t-1} - \alpha \frac{m_{t_{\text{hat}}}}{\sqrt{v_{t_{\text{hat}}}} + \epsilon} \tag{6}$$

where,

- β_1 and β_2 are the exponential decay rates for the first and second moment estimates, respectively.
- α is the learning rate.
- $\nabla_\theta L(\theta_{t-1}$ represents the gradient of the loss function with respect to the parameters at time step t-1.
- ϵ is a small value added for numerical stability.

On the other hand, the loss function chosen for our model is categorical cross-entropy. Categorical cross-entropy is commonly used in multiclass classification tasks. It measures the dissimilarity between the predicted probability distribution and the true probability distribution of the target classes. By minimizing the categorical cross-entropy loss, our model aims to optimize the predictions for accurate class probabilities.

Mathematical expression of categorical cross-entropy:

$$L(y_{\text{true}}, y_{\text{pred}}) = -\sum y_{\text{true}} \log(y_{\text{pred}}) \tag{7}$$

In this equation, y_{true} represents the true probability distribution of the target classes, typically in the form of a one-hot encoded vector. y_{pred} represents the predicted probabilities for each class, obtained from the model's output. The expression $[y_{\text{true}} \log(y_{\text{pred}}]$ calculates the element-wise product of the true probabilities and the logarithm of the predicted probabilities. The negative sign ensures that the loss is minimized during optimization.

Additionally, Callback parameters, such as automatic learning rate reduction and early stopping, are also specified. The model is then trained using the fit function, utilizing the training dataset, and validated using a separate validation dataset. After training, the model is saved in the .h5 format for future use.

The Adam optimizer offers clear advantages for multiclass classification tasks due to its adaptive learning rate and momentum-based updates. Its ability to adjust learning rates for different parameters is essential given the varying complexities and importance of features and classes. Furthermore, the momentum-driven updates help Adam navigate complex optimization landscapes and avoid being trapped in suboptimal solutions. The optimizer's inclusion of bias correction during initial iterations further accelerates convergence and improves accuracy, particularly beneficial when dealing with intricate decision boundaries in multiclass settings. So, Adam optimizer is selected for model optimization. The effect of dynamic learning rate adjustment can be visualized inFigure 5 The model is initially unstable in the early training stages because of a high learning rate(default learning rate used i.e 0.001), but it becomes more stable as the learning rate is dynamically adjusted.

To assess the model's performance, accuracy and loss graphs are plotted using Matplotlib. Additionally, the model is evaluated using parameters such as precision, recall, and F1 score using confusion matrix which is describe on section IV, providing further insights into its effectiveness.

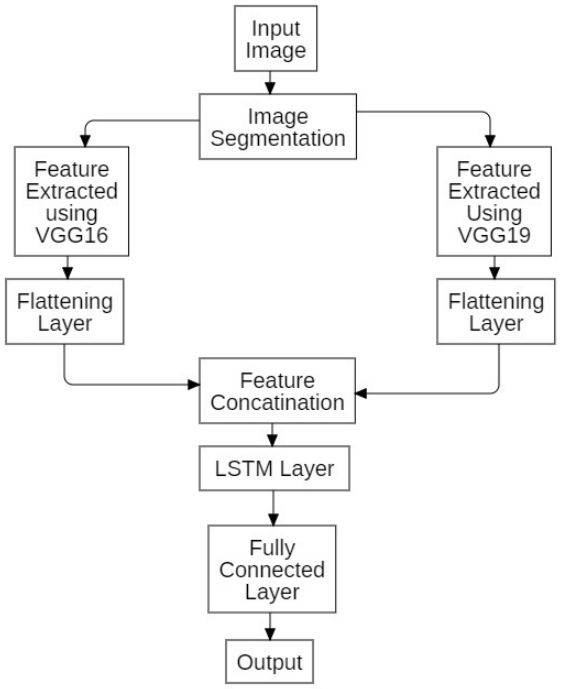


Fig. 3. System Workflow

C. Model Architecture

Model: VGG16-VGG19-LSTM Model			
Layer	**Output Shape**	**Param #**	**Connected to**
input_1 (InputLayer)	(None, 256, 256, 3)	0	
model (Functional)	(None, 8, 8, 512)	14,714,688	input_1[0][0]
model_1 (Functional)	(None, 8, 8, 512)	20,024,384	input_1[0][0]
flatten (Flatten)	(None, 32768)	0	model[0][0]
flatten_1 (Flatten)	(None, 32768)	0	model_1[0][0]
concatenate (Concatenate)	(None, 65536)	0	flatten[0][0], flatten_1[0][0]
lambda (Lambda)	(1, 128, 512)	0	concatenate[0][0]
lstm (LSTM)	(1, 128)	278,784	lambda[0][0]
dense (Dense)	(1, 4)	260	dropout[0][0]
Total params: 35,067,780			
Trainable params: 328,708			
Non-trainable params: 34,739,072			

TABLE III
INTEGRATED MODEL SUMMARY

The model architecture shown in the figure above is a combination of two CNN models: VGG16 and VGG19, along with an LSTM model and a fully connected dense layer. The variable "model" represents the VGG16 model, while "model_1" represents the VGG19 model. Both models are connected to a single input layer (input_1) to receive the same image as input. The output of "model" (VGG16) is connected to a flatten layer, and the output of "model_1" (VGG19) is connected to flatten_1 layer. These flatten layers are then concatenated together. Using a lambda layer, the flattened output is reshaped to be input into the LSTM model. The LSTM model consists of 128 memory cells, where the value of each time step is stored in a single memory cell. The lambda layer reshapes the flattened output in such a way that a sequence of pixelated features, with 512 features of the same index, is provided as input to determine the pixel-wise dependency among the extracted features.

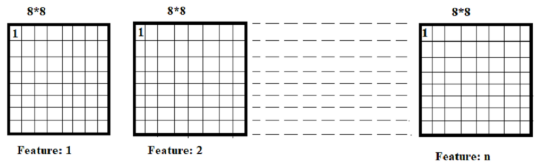


Fig. 4. Feature Converted to Sequential Data

So, there are a total of 64 time steps for each CNN model (VGG16 and VGG19). Each time step corresponds to a pixel feature in the image. The included figureFigure 4 visually illustrates how the pixel features are transformed into sequential data for the LSTM model. For each time step, the feature data is inputted in the following format:

$$Sequence1 : Feature1.1, Feature2.1, ..., Featuren.1 \quad (8)$$

This pattern continues for the remaining 63 sequences, where each sequence represents a different pixel index. Since we are using two CNN models (VGG16 and VGG19), there are a total of 128 time steps in our model. This means there are 64 time steps for VGG16 and another 64 time steps for VGG19. Each time step captures the features corresponding to a specific pixel index.

Finally, the LSTM model is connected to a dense layer for the final prediction. The model achieved a final accuracy of 95.50%.

Given below are the hyperparameter changes performed to increase our model accuracy.

VGG_16 + VGG_19 + LSTM							
SN	**Hyperparameters**	**Test1**			**Test2**		
		Train Acc	**Test Acc**	**Epoch**	**Train Acc**	**Test Acc**	**Epoch**
"8" different real-time data augmentation were performed							
rescale: 1/255, horizontal_flip, vertical_flip, rotation_range: 30, zoom_range: 0.2, width_shift_range: 0.1, height_shift_range: 0.2, shear_range: 0.2							
1	64 memory cell	83.89%	86.50%	19	87.94%	88.50%	40
2	128 memory cell	89.09%	88.83%	46	89.29%	89.67%	39
3	256 memory cell	89.52%	89.17%	40	89.15%	88.67%	45
Real time data augmentation was reduced to "6"							
rescale: 1/255, rotation_range: 30, zoom_range: 0.2, width_shift_range: 0.1, height_shift_range: 0.2, shear_range: 0.2							
4	128 memory cell	92.22%	93.17%	52	93.40%	93.50%	49
Real time data augmentation was reduced to "4"							
rescale: 1/255, rotation_range: 30, zoom_range: 0.2, shear_range: 0.2							
5	128 memory cell	92.99%	93.17%	40	95.42%	95.50%	36

TABLE IV
HYPER PARAMETERS CHANGES EXPERIMENTS OF INTEGRATED MODEL

IV. RESULT AND DISCUSSION

Various approaches were taken for the evaluation of our integrated model.

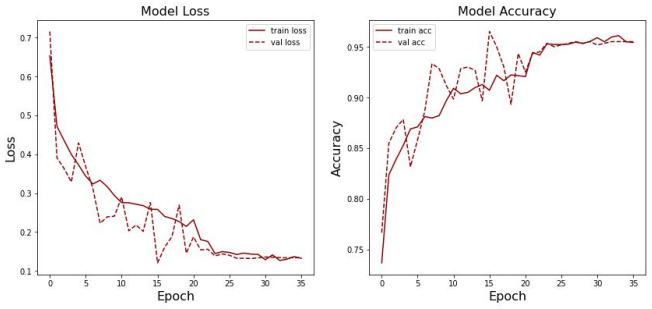


Fig. 5. Loss and Accuracy Plot Of Integrated Model

The final result can be observed in the table given below.

Final Train Accuracy	95.42%
Final Test Accuracy	95.50%
Final Train Loss	0.1321
Final Test Loss	0.1338

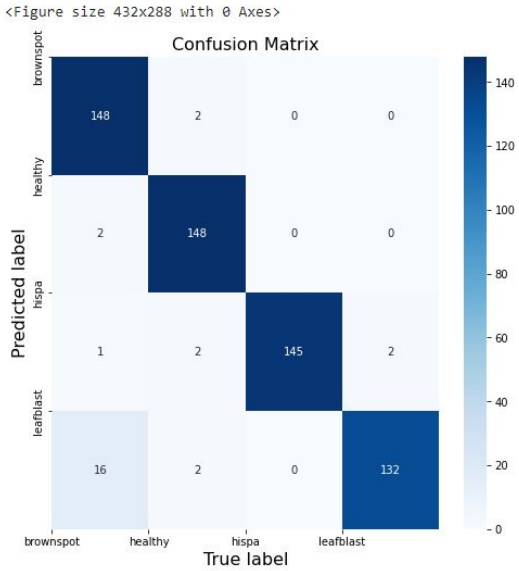


Fig. 6. Confusion Matrix Of Integrated Model

In addition to comparing the final training and testing accuracy and loss, the article also includes the plotting of a confusion matrix. Which is a valuable tool for assessing the performance of a model by determining the number of True positive (TP), False positive (FP), and False negative (FN) instances.

By using the values from the confusion matrix, we calculated additional evaluation metrics such as Precision (Positive Predictive Value), Recall (Sensitivity), F-Measure, and AUC (Area Under ROC). Analyzing the confusion matrix and calculating these metrics allows us to assess the model's precision in correctly identifying positive cases, its recall in capturing all positive cases, and overall effectiveness in terms of accuracy and predictive power.

A. Precision

Precision measures of the accuracy of positive predictions or decisions made by a model or a system. Mathematically, it can be expressed as:

$$Precision = \frac{TP}{(TP + FP)} \tag{9}$$

where:

TP = True Positive

It is the number of cases where the model correctly identifies a sample as belonging to a specific class when it does indeed belong to that class.

FP = False Positive

It is the number of case where the model mistakenly identifies a sample as belonging to a specific class when it actually belongs to a different class.

B. Recall

Recall measures how well the system is able to identify all instances of the positive class. Mathematically, it can be expressed as:

$$Recall = \frac{TP}{(TP + FN)} \tag{10}$$

where:

TP = True Positive

It is the number of cases where the model correctly identifies a sample as belonging to a specific class when it does indeed belong to that class.

FN = False Negative

It is the number of case where the model mistakenly identifies a sample as not belonging to a specific class when it actually does belong to that class.

C. F1 Score

The F1 score is a statistical metric used to evaluate the performance of a classification or detection system. Mathematically, it can be expressed as:

$$F1Score = \frac{2 * (precision * recall)}{(precision + recall)} \tag{11}$$

Classes	**Brownspot**	**Healthy**	**Hispa**	**Leafblast**
Precision score per class	0.8862	0.9610	1.0	0.9851
Recall score per class	0.9867	0.9867	0.9667	0.88
F1 score per class	0.9338	0.9737	0.9830	0.9296

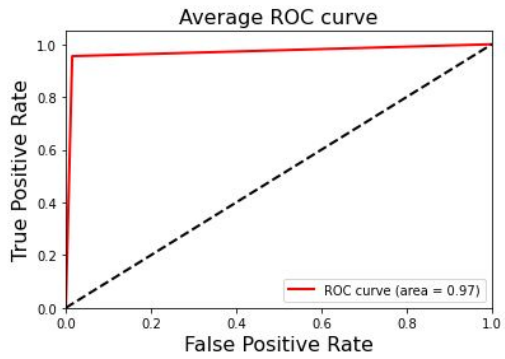


Fig. 7. Precision-Recall plot and Average_roc_curve Of Integrated Model

SN	Model	Original Image			Segmented Image		
		Train ACC	Test ACC	Epoch	Train ACC	Test ACC	Epoch
1	VGG16_ VGG19_ LSTM	91.11%	93.17%	75	95.42%	95.50%	36
2	VGG16	86.15%	80%	22	88.24%	87.83%	44
3	VGG19	84.70%	74%	41	87.26%	88.83%	66
4	Densenet_ 121	88.85%	91.83%	31	81.77%	84.50%	26
5	VGG16_ LSTM	88.75%	91.50%	30	86.76%	88.50%	31
6	VGG19_ LSTM	85.58%	84.83%	25	86.73%	89.33%	36

TABLE V
MULTI CLASS CLASSIFICATION MODEL COMPARISION

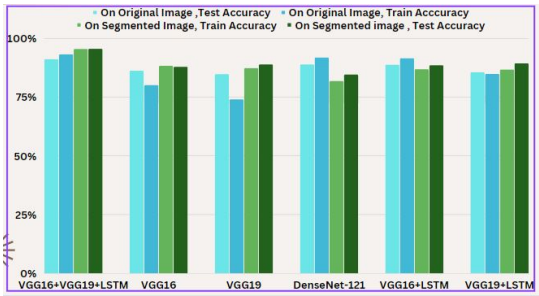


Fig. 8. Graph of Multi Class Classification Model Comparision

By analyzing the graph, we can observe a significant difference between the training and testing accuracy when training the different models on the original image dataset. This difference indicates the presence of overfitting and underfitting conditions, where the model is either too complex or too simple to accurately generalize the data.

However, when training the images on a segmented image dataset, we observe only a small variation between the training and testing accuracy. This suggests that the process of segmentation helps in achieving more stable training of the dataset. The segmented dataset allows the model to focus specifically on the diseased parts of the images, which can lead to improved generalization and reduced overfitting or underfitting issues.

V. CONCLUSION AND FUTURE ENHANCEMENT

The main emphasis of the article is on the improvement achieved by incorporating the LSTM model into the CNN model and the significance of image segmentation in image classification. Referring to the comparison of models presented in Table V, We can observe that, in general, the training graph of our model trained on the segmented images exhibits greater stability compared to the model trained on the original images. This distinction can be visualized in the training graph depicted in Figure 5.

As for Future Enhancements to our project, we could implement these:

- More disease classes can be added.
- Improvements can be made to the segmentation quality.
- Multi-disease classification in a single leaf image can be implemented.

REFERENCES

[1] D. Garg and M. Alam, "Integration of convolutional neural networks and recurrent neural networks for foliar disease classification in apple trees," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4, 2022.

[2] P. Li, H. Tang, J. Yu, and W. Song, "Lstm and multiple cnns based event image classification," *Multimedia Tools and Applications*, vol. 80, pp. 30 743–30 760, 2021.

[3] T. Islam, M. Sah, S. Baral, and R. R. Choudhury, "A faster technique on rice disease detectionusing image processing of affected area in agro-field," in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*. IEEE, 2018, pp. 62–66.

[4] M. Z. Islam, M. M. Islam, and A. Asraf, "A combined deep cnn-lstm network for the detection of novel coronavirus (covid-19) using x-ray images," *Informatics in medicine unlocked*, vol. 20, p. 100412, 2020.

[5] A. K. Rangarajan, R. Purushothaman, and A. Ramesh, "Tomato crop disease classification using pre-trained deep learning algorithm," *Procedia computer science*, vol. 133, pp. 1040–1047, 2018.

[6] R. Karthik, M. Hariharan, S. Anand, P. Mathikshara, A. Johnson, and R. Menaka, "Attention embedded residual cnn for disease detection in tomato leaves," *Applied Soft Computing*, vol. 86, p. 105933, 2020.

[7] S. Lamba, A. Baliyan, and V. Kukreja, "A novel gcl hybrid classification model for paddy diseases," *International Journal of Information Technology*, pp. 1–10, 2022.

[8] N. Krishnamoorthy, L. N. Prasad, C. P. Kumar, B. Subedi, H. B. Abraha, and V. Sathishkumar, "Rice leaf diseases prediction using deep neural networks with transfer learning," *Environmental Research*, vol. 198, p. 111275, 2021.

[9] M. Al-Amin, D. Z. Karim, and T. A. Bushra, "Prediction of rice disease from leaves using deep convolution neural network towards a digital agricultural system," in *2019 22nd International Conference on Computer and Information Technology (ICCIT)*. IEEE, 2019, pp. 1–5.