

**Sentinel Harvest Watcher: An App for Rice Disease Detection in Complex Backgrounds using PSO-CNN**

Anshul Anilkumar Mundakatil

017416334

San Jose State University

Final Project Report

ISE 244: AI Tools and System Engineering

## Problem Definition

Crops play a vital role in human life, and their growth and sustainability are crucial in the long run. Rice, especially, is a key agricultural crop; however, it often suffers from various diseases, leading to decreased yields and, in severe cases, crop failure. Diseases significantly impact rice growth and yield, resulting in economic losses and food security challenges. Image recognition plays a critical role in identifying rice diseases, facilitating automated and efficient detection, essential for effective management, ensuring food security, and promoting sustainable agriculture.

Over the years, machine learning and artificial intelligence have experienced tremendous growth, and everyone seeks to leverage their power. One of the best ways to bring AI to agriculture is through Computer Vision. Computer Vision, a field in Artificial Intelligence, aims to automate tasks that the human visual system can perform, such as recognizing objects, navigating environments, and understanding scenes.

In this project, we aim to address the complex challenges involved in rice disease detection. One of the major challenges in crop disease detection is the complex background. The images used in training are of leaves with no background (Taimur Ahad et al., 2002), which makes training easier for the model. However, this increases complexity for users, as they need to ensure that the background does not contain noise, which is unlikely on farms, where fields are vast. Expecting users to pluck a yield, place it on a clear surface, and use the app to determine whether it has a disease is impractical.

Convolutional Neural Networks (CNNs) provide exceptional results (Wang et al., 2021); however, since they are typically not trained on complex backgrounds, they tend to underperform in practical applications. Other algorithms used in recent studies show impressive results (Ahmed et al., 2019), but CNNs outperform them. Hasan et al. (2019) used an SVM classifier in conjunction with an AI model named deep CNN, then retrained the model using 1080 images from nine different rice disease datasets. The features extracted from the DCNN model were then used to train the SVM classifier. This model proficiently recognized and classified nine different types, achieving an accuracy rate of 97.5% for rice disease classification, but the DCNN model was prone to overfitting and had slow training speeds.

Using CNNs is the optimal solution, but knowing which hyperparameters to choose to train images with complex backgrounds correctly is challenging. To address this, we propose a novel approach, PSO-CNN (Particle Swarm Optimization Convolution Neural Network). The Particle Swarm Optimization approach (Kennedy et al., 1995) can be used to find the best hyperparameters for CNNs (pyswarm). The goal of this project is to develop a mobile-based

application that can accurately determine the condition of the crop. To convert the powerful model into model that can be used on mobile phones, we leverage TensorFlow Lite. TensorFlow Lite (TensorFlow) is a set of tools that enables on-device machine learning, helping developers run their models on mobile, embedded, and edge devices.

This project aims to revolutionize rice disease detection in agriculture by leveraging Computer Vision and AI technologies. Through the development of the PSO-CNN model and its adaptation for mobile devices using TensorFlow Lite, we strive to address the complex challenges posed by rice diseases, ultimately contributing to sustainable agriculture practices, and ensuring food security.

## Project Objectives

The objective of this project is to develop a robust application that can accurately detect various rice diseases. We will begin by outlining the classification requirements for our project.

### I. Classification Objective



The classification requirement for this project is to identify the disease class present in the plant. The disease classes include 'Bacterial Leaf Blight', 'Brown Spot', 'Healthy Rice Leaf', 'Leaf Blast', 'Leaf scald', and 'Sheath Blight'.

**Table 1** summarizes the rice disease dataset, providing information on the categories, classes, and the number of images in each category. This table offers a clear overview of the image distribution across the different rice disease classes.







### II. Image Dataset Requirements

The next objective of the project is to classify images under complex backgrounds. To achieve this, we need to find datasets that contain noise (refer to **Table 2**, Sample 2). The most commonly available datasets are those with no background (refer to **Table 2**, Sample 1). To address this, we obtained the rice disease dataset by sourcing images from the Kaggle platform. The Rice Leaf Diseases Detection dataset was used for the project, leveraging a total of 3829 images, with each class containing approximately 600-plus images.

**Table 2**

Sample	Example	Description
Sample 1		Commonly available datasets have backgrounds with no noise, making them unsuitable for practical scenarios.
Sample 2		Expected Dataset with background noise as this makes it ideal for practical scenario.

**Table 1**

Category	Example	Characteristics	Number
Bacterial Leaf Blight		Greenish yellowish-white spots appearing at the base of the heart leaves, later expanding into yellow stripes parallel to the leaf veins, remaining green between the stripes.	636
Brown Spot		Brown Spot is a common fungal disease in rice characterized by small, dark brown spots with yellow halos on the leaves, often leading to reduced photosynthesis and yield	646
Healthy Rice leaf		Robust, resilient seedling with broad, sturdy leaves, short leaf sheaths and thick, flat pseudo stems.	653
Leaf Blast		Leaf blast in rice is characterized by elliptical or spindle-shaped lesions with gray centers and dark borders on the leaves. These lesions can coalesce, leading to extensive damage and reduced photosynthetic capacity in the affected plants.	634
Leaf Scald		Leaf scald is a vascular disease caused by the bacterium <i>Xanthomonas albilineans</i> , characterized by creamy or grayish streaking and later withering of the leaves	628
Sheath Blight		The fungus affects the crop from tillering to heading stage. Initial symptoms are noticed on leaf sheaths near water level. On the leaf sheath oval or elliptical or irregular greenish grey spots are formed. As the spots enlarge, the center becomes greyish white with an irregular blackish brown or purple, brown border.	632

### **III. Developing a Deep Learning Model**

Convolutional Neural Networks (CNNs) have shown exceptional results (Wang et al., 2021). However, they tend to underperform in practical applications when not trained on complex backgrounds. To address this challenge and find the best parameters for training on images with complex backgrounds, we propose PSO-CNN, a novel approach suggested in this project. PSO-CNN identifies the optimal parameters, which can then be used to train a fully connected CNN.

### **IV. Saving Complex Deep Learning Models**

Once our model is developed, we need to save our deep learning model for future use. We have several options to choose from. Among these options, we will select TensorFlow Lite. TensorFlow Lite models are optimized for mobile hardware, providing high performance and efficiency for on-device machine learning applications.

### **V. Developing a Mobile Application**

Since mobile phones are widely used and accessible to the general consumer, we will develop an application specifically for mobile phones. For application development, we will use Android Studio, a powerful tool that will help us build user-friendly applications with integrated deep learning models.

## Analysis

**Figure 1** shows a detailed system architecture. We will analyze each component of the system architecture.

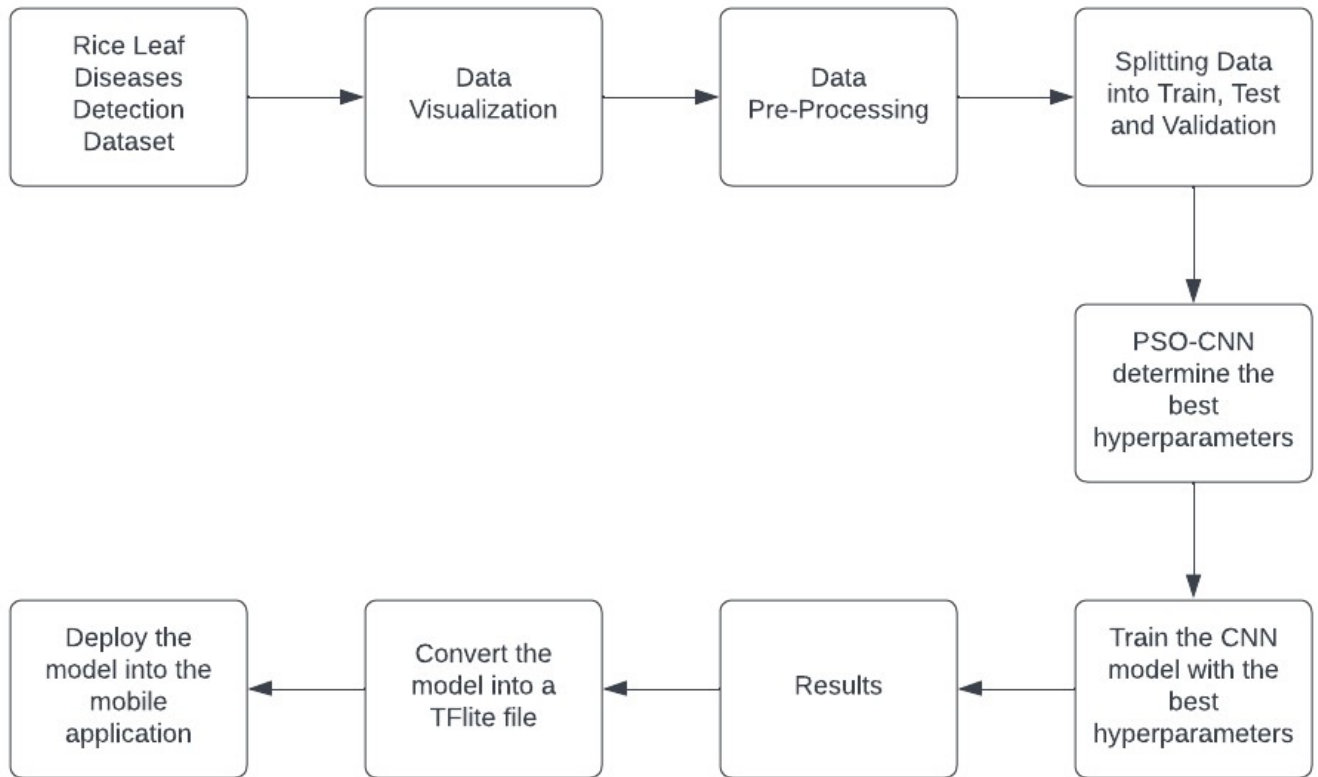


Figure 1

### I. Rice Disease Dataset

The Rice Leaf Diseases Detection Dataset is a collection of images featuring rice leaves in diverse conditions, encompassing both healthy and diseased states. This dataset is curated for the purpose of detecting and classifying distinct types of rice leaf diseases: Bacterial Leaf Blight, Brown Spot, Healthy Rice Leaf, Leaf Blast, Leaf scald, and Sheath Blight. Each category represents a specific pathology affecting rice plants. A total of 3829 images were used from the dataset, with each class having approximately 600 images.

### II. Data Visualization

**Figure 2** visualizes the structure of the dataset. We can see that the distribution is even, as we aim to avoid overfitting the model. **Figure 3** shows the visual representation of the dataset images with their labels.

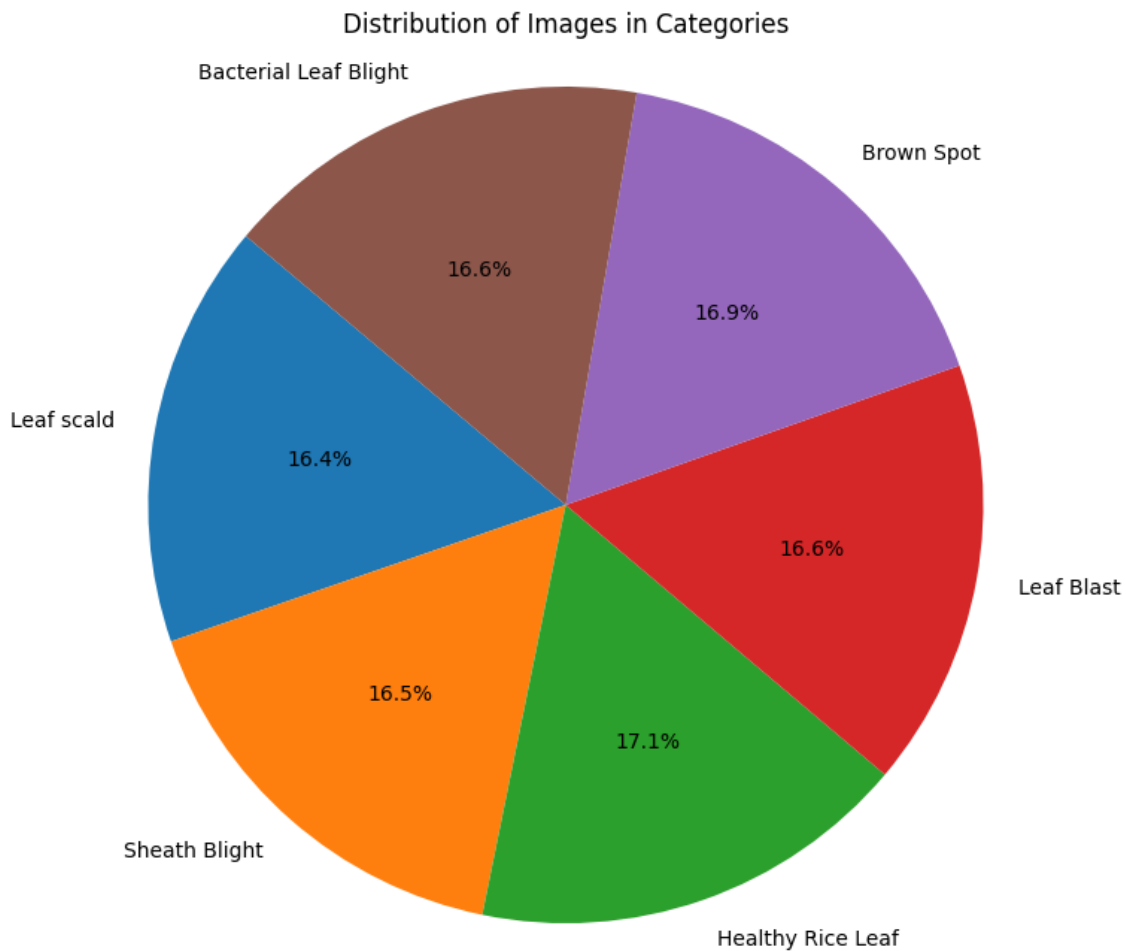


Figure 2

### III. Data Pre-Processing

For data pre-processing, we resize the images to 128. We do not perform data augmentation as our data is already extensive and includes augmented images as well.

### IV. Splitting the Data

We split the data into training, testing, and validation sets with the following distribution:

- 80% of the data is allocated to the training set.
- 10% of the data is allocated to the validation set.
- 10% of the data is allocated to the testing set.

We also shuffle the dataset before splitting, which is useful for randomizing the order of examples, especially important in scenarios where the data might be ordered in some meaningful way (like time series data).





Figure 3

## V. PSO-CNN (Particle Swarm Optimization - Convolutional Neural Network)

### A. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is explained beautifully in the Pyswarms documentation titled (It's all a treasure hunt) which narrates the concept as a captivating story.

“Imagine that you and your friends are looking for a treasure together. The treasure is magical, and it rewards not only the one who finds it, but also those near to it. Your group knows, approximately, where the treasure is, but is not exactly sure of its definite location.

Your group then decided to split up with walkie-talkies and metal detectors. You use your walkie-talkie to inform everyone of your current position, and the metal detector to check your proximity to the treasure. In return, you gain knowledge of your friends' positions, and their distance from the treasure.

As a member of the group, you have two options:

1. Ignore your friends, and just search for the treasure the way you want it. The problem is, if you didn't find it, and you're far away from it, you get a very low reward.
2. Using the information you gather from your group, coordinate, and find the treasure together. The best way is to know who is the one nearest to the treasure and move towards that person.

Here, it is evident that by using the information you can gather from your friends, you can increase the chances of finding the treasure, and at the same time maximize the group's reward. This is the basics of Particle Swarm Optimization (PSO). The group is called the swarm, you are a particle, and the treasure is the global optimum [CI2007]."

In this project, we utilize Particle Swarm Optimization (PSO) to optimize hyperparameters for a Convolutional Neural Network (CNN) designed for image classification with background noise. PSO is employed to determine the optimal values for the number of neurons in each layer of the CNN. The PSO algorithm is run for 10 epochs to find the best parameters for CNN.

### ***B. Pseudo Code***

Algorithm: Train Network using PSO

Input: hyperparameters

Output: scores

1. Begin
2. Extract neuron\_1, neuron\_2, neuron\_3 from hyperparameters
3. Create a neural network model psomodel with the following layers:
  - Rescaling layer with scale 1./255
  - Convolutional layer with neuron\_1 filters, 3x3 kernel, and 'relu' activation
  - MaxPooling layer
  - Convolutional layer with neuron\_2 filters, 3x3 kernel, and 'relu' activation

- MaxPooling layer
  - Convolutional layer with neuron\_3 filters, 3x3 kernel, and 'relu' activation
  - MaxPooling layer
  - Flatten layer
  - Dense layer with 128 units and 'relu' activation
  - Dense layer with 6 units (output layer)
4. Compile the psomodel with 'adam' optimizer, SparseCategoricalCrossentropy loss, and 'accuracy' metric
  5. Train the psomodel on training data train\_ds, with validation data val\_ds, verbose=1, and 10 epochs
  6. Evaluate the psomodel on test data test\_ds, store the scores
  7. Return negative of the scores
  8. Run the PSO model with upper bound and lower bound limits of neurons set between [10, 100]
  9. Get the best parameters.
  10. End

The PSO algorithm successfully provides the best hyperparameters for the CNN, resulting in improved performance in classifying images with background noise.

## VI. Convolution Neural Network

The model **Figure 4** starts with a rescaling layer, which ensures that the input pixel values are in the range [0, 1]. This is important for neural networks to process the data effectively.

Then, the model consists of three sets of convolutional and max pooling layers. These layers are responsible for extracting features from the input images. The number of layers is determined by PSO-CNN, which selects the optimal configuration for our CNN.

The convolutional layers apply filters to the input image, which helps in detecting patterns and features at different spatial locations. The max pooling layers reduce the spatial dimensions of the feature maps, which helps in reducing the computational complexity of the model.

After the convolutional and max pooling layers, the model has a flatten layer, which converts the 3D output from the previous layers into a 1D vector. This flattened vector is then fed into a series of dense (fully connected) layers.

The dense layers learn to classify the features extracted by the convolutional layers into different classes. The final dense layer uses softmax activation to produce probabilities for each class, making it suitable for multi-class classification tasks.

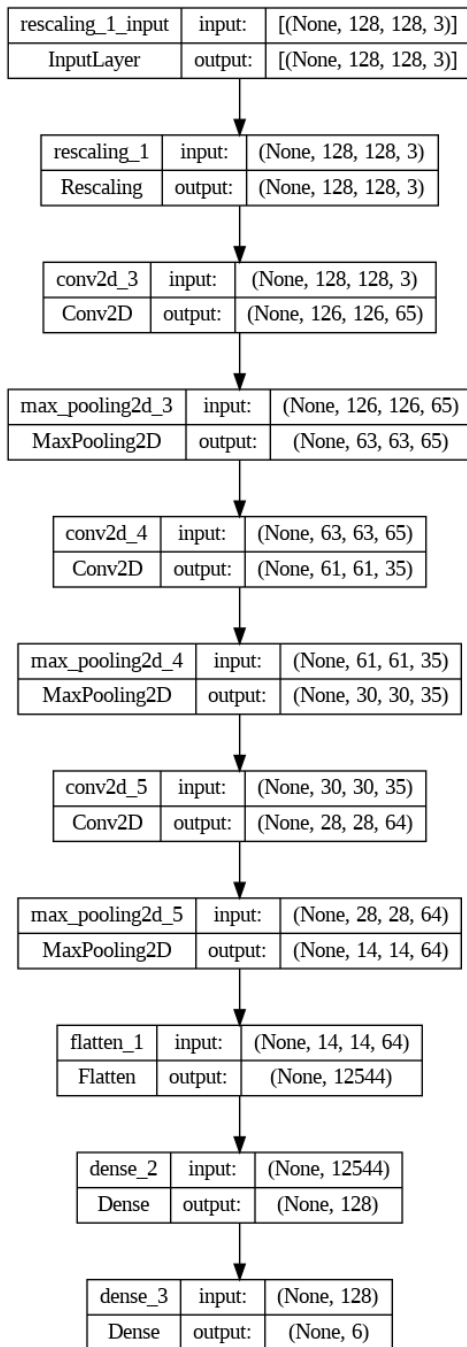


Figure 4

Table 3

Layers (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 128, 128, 3)	0
conv2d_3 (Conv2D)	(None, 126, 126, 65)	1820
max_pooling2d_3 (MaxPooling2D)	(None, 63, 63, 65)	0
conv2d_4 (Conv2D)	(None, 61, 61, 35)	20510
max_pooling2d_4 (MaxPooling2D)	(None, 30, 30, 35)	0
conv2d_5 (Conv2D)	(None, 28, 28, 64)	20224
max_pooling2d_5 (MaxPooling2D)	(None, 14, 14, 64)	0
flatten_1 (Flatten)	(None, 12544)	0
dense_2 (Dense)	(None, 128)	1605760
dense_3 (Dense)	(None, 6)	774
Total params: 1649088 (6.29 MB)		
Trainable params: 1649088 (6.29 MB)		
Non-trainable params: 0 (0.00 Byte)		

## VII. Result Metrics

We have employed various methods to visualize and understand our results. We begin by plotting Training and Validation Metrics. Next, we visualize Model Predictions on Test Images. Finally, we plot the confusion matrix and classification report.

### ***A. Training and Validation Metrics***

The training and validation metrics provide insights into how well our model is learning from the training data and generalizing to unseen validation data.

### ***B. Model Predictions on Test Images***

Visualizing model predictions on test images helps us understand how well the model is performing on real-world data. We display the actual test images along with the model's predicted labels to evaluate its accuracy and ability to correctly classify different rice leaf diseases.

### ***C. Confusion Matrix***

The confusion matrix is a useful tool for understanding the performance of a classification model. It provides a summary of correct and incorrect predictions, highlighting any patterns or areas where the model may be struggling.

### ***D. Classification Report***

The classification report provides a detailed breakdown of metrics such as precision, recall, and F1-score for each class in the dataset. This report gives us a comprehensive understanding of how well the model is performing for each disease category.

## VIII. Conversion to TFLite Model

We utilize `TFLiteConverter.from_keras_model` (see **Figure 5**) to convert our model into a TFLite model.

## IX. Mobile Application Development

For developing our mobile application, we employ Android Studio. Android Studio offers a plethora of powerful features, and deploying our TFLite model is straightforward; we simply add it as our machine learning model (see **Figure 6**).

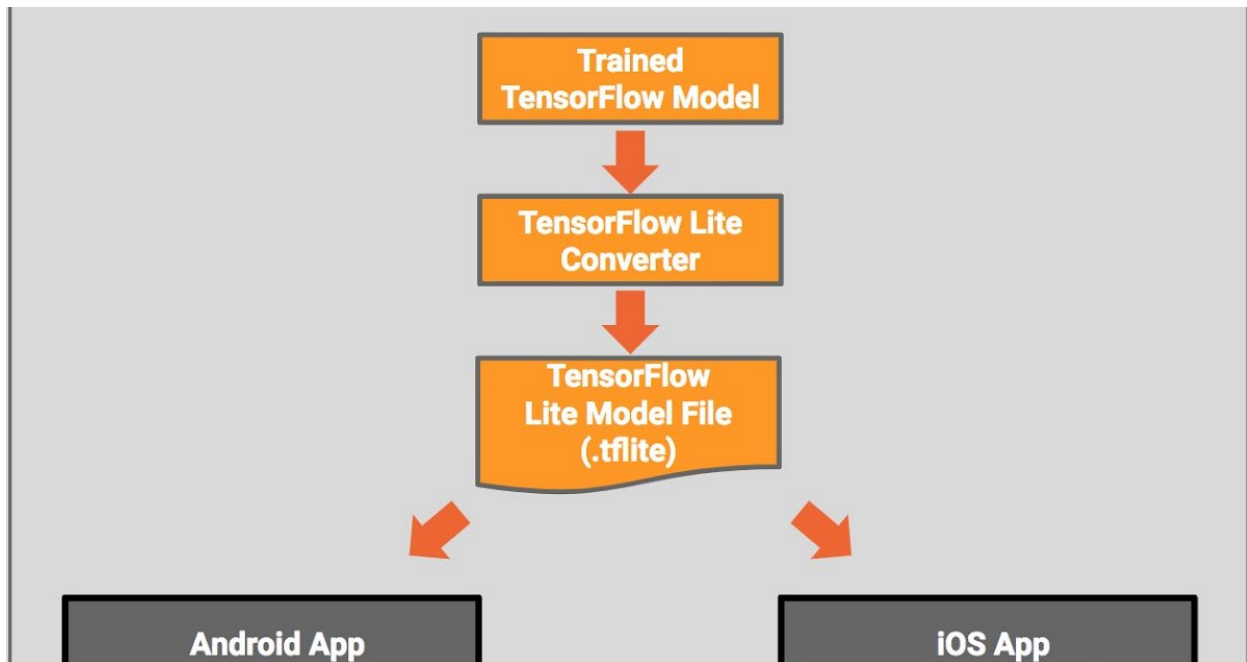


Figure 5

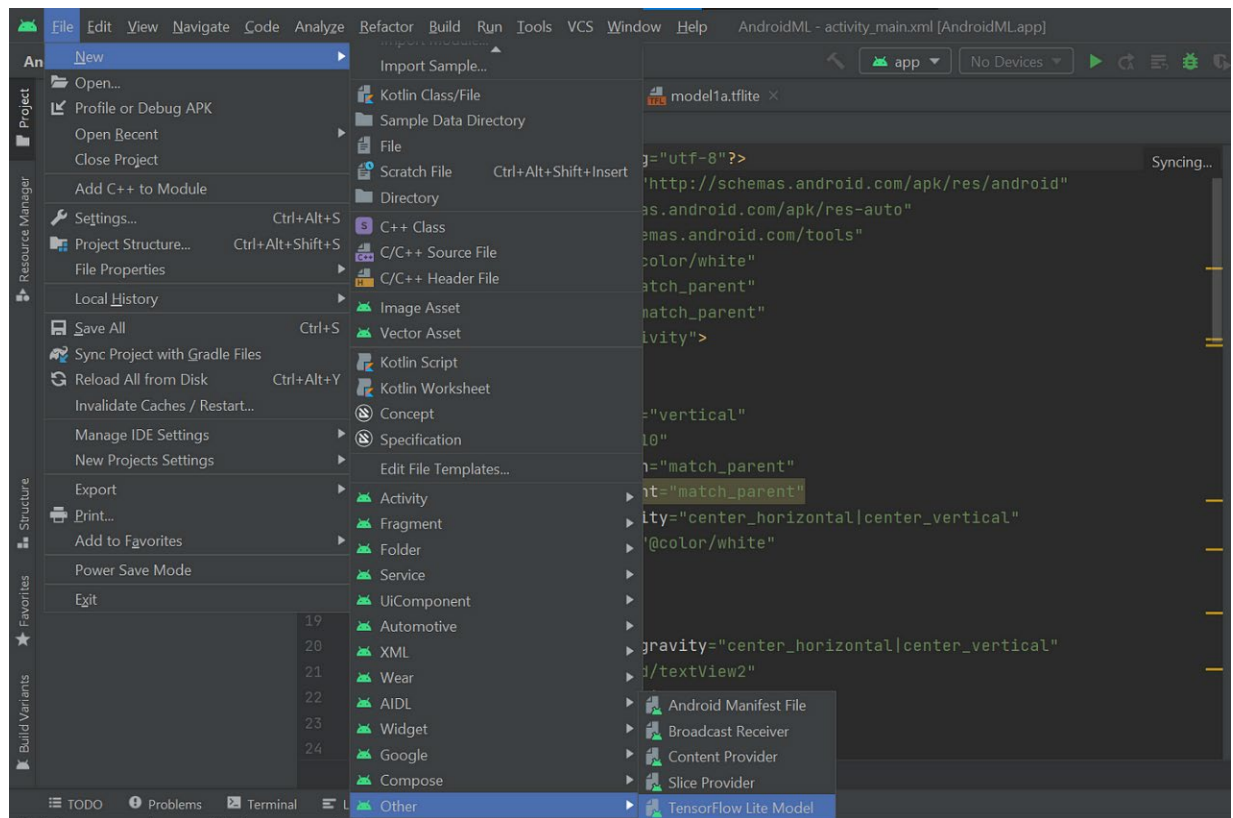


Figure 6



## Results

In this project, we have used multiple metrics to assess the performance of our model. Each of these metrics gives us an idea of how well the model is performing.

### I. Model Predictions on Test Images

We use visualizations of the model's predictions **Figure 8** on test images to assess its performance with data. This involves presenting the actual test images alongside the model's predicted labels, allowing us to evaluate its accuracy and its ability to correctly classify various rice leaf diseases.

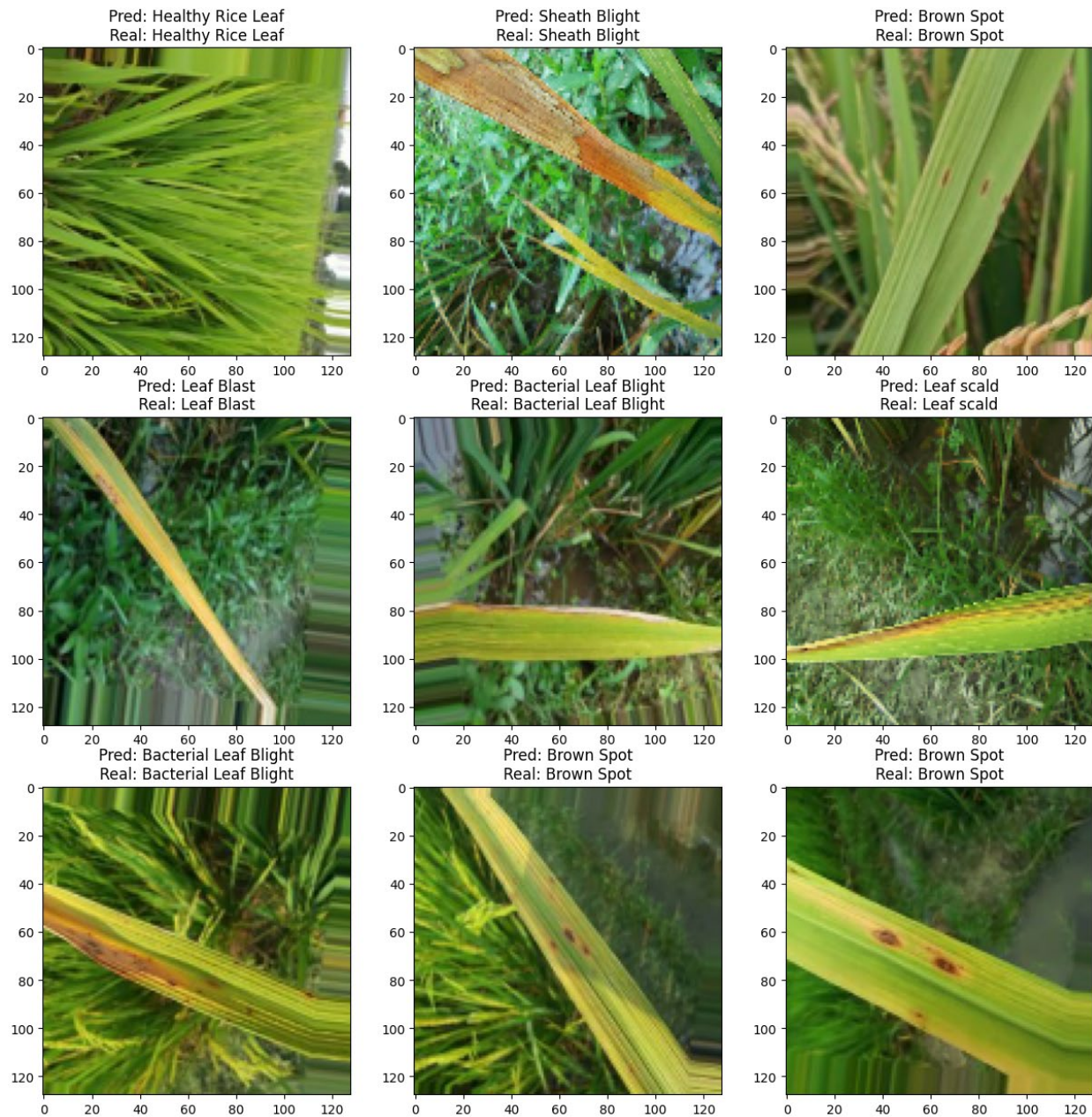


Figure 7

## II. Confusion Matrix

A confusion matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is a means of displaying the number of accurate and inaccurate instances based on the model's predictions. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance. The matrix displays the number of instances produced by the model on the test data.

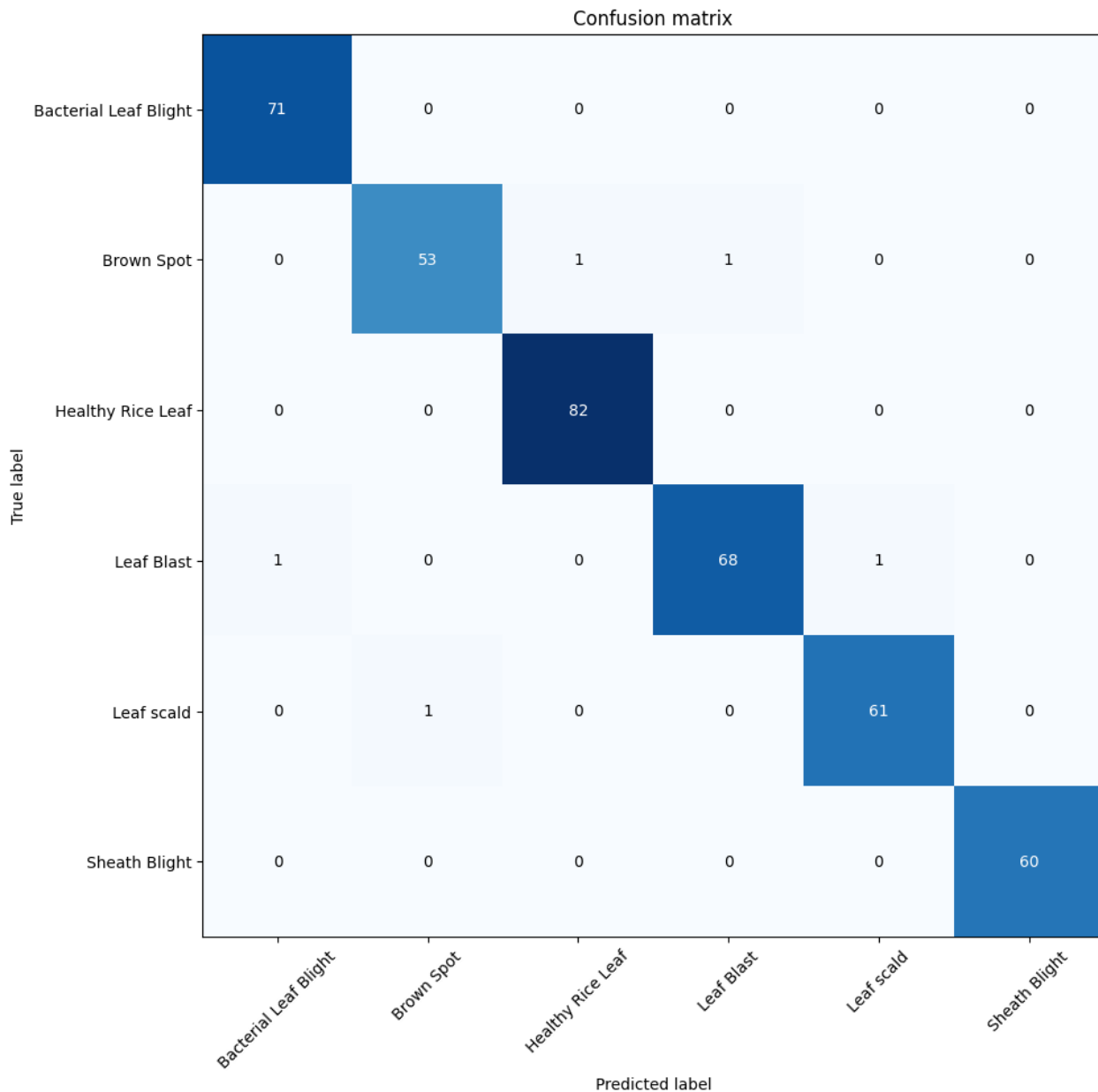


Figure 8

Here (**Figure 8**), we have plotted the true labels on the y-axis and the predicted labels on the x-axis. As seen from the confusion matrix, our model performs quite well with the test data. Although it does predict a few instances incorrectly, overall, it is a strong model.



### III. Classification Report

Table 4

Class	Precision	Recall	F1-Score	Support
Bacterial Leaf Blight	0.99	1.00	0.99	71
Brown Spot	0.98	0.96	0.97	55
Healthy Rice Leaf	0.99	1.00	0.99	82
Leaf Blast	0.99	0.97	0.98	70
Leaf scald	0.98	0.98	0.98	62
Sheath Blight	1.00	1.00	1.00	60
Accuracy			0.99	400
Macro Avg	0.99	0.99	0.99	400
Weighted Avg	0.99	0.99	0.99	400

This classification report provides a detailed evaluation of a machine learning model's performance on a dataset containing different classes of rice leaf diseases. Here's a breakdown of the key metrics:

#### A. Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positives. For example, for Bacterial Leaf Blight, the precision is 0.99, indicating that 99% of the samples predicted as Bacterial Leaf Blight were Bacterial Leaf Blight.

#### B. Recall

Recall is the ratio of correctly predicted positive observations to all actual positives. For Bacterial Leaf Blight, the recall is 1.00, meaning the model correctly identified all instances of Bacterial Leaf Blight in the dataset.

#### C. F1-score

The F1-score is the weighted average of precision and recall. It is a measure of a test's accuracy, with the best score being 1.0 and the worst 0.0. It indicates the balance between precision and recall. For Bacterial Leaf Blight, the F1-score is 0.99, indicating high accuracy.

#### D. Support

Support is the number of actual occurrences of the class in the specified dataset. For example, there are 71 instances of Bacterial Leaf Blight in the dataset.

### ***E. Accuracy***

Overall accuracy of the model, which is the ratio of correctly predicted instances to the total instances in the dataset. Here, the accuracy is 0.99, indicating that the model correctly predicted 99% of the instances in the dataset.

### ***F. Macro Avg***

The average precision, recall, and F1-score across all classes. Here, it's 0.99, indicating high performance across all classes.

### ***G. Weighted Avg***

The weighted average of precision, recall, and F1-score, considering the number of instances of each class. Here, it's also 0.99, indicating that the model performs well when considering the class imbalance in the dataset.

## Discussion

Discussing our results further, we observe that our PSO-CNN performs extremely well under complex backgrounds. We addressed two major challenges: handling cropped images with complex backgrounds and determining the best hyperparameters using PSO-CNN

Training a Convolutional Neural Network (CNN) with images containing complex backgrounds can be beneficial over using images with plain backgrounds for several reasons

### ***A. Real-world scenario simulation***

Images with complex backgrounds more closely resemble real-world scenarios where objects of interest are surrounded by various elements. Training on such images helps the model learn to distinguish between the object and the background, improving its ability to generalize to new, unseen images.

### ***B. Robustness***

CNNs trained on complex backgrounds tend to be more robust to variations in backgrounds and lighting conditions. This robustness can result in better performance when the model is deployed in diverse environments.

### ***C. Feature extraction***

Complex backgrounds provide a richer set of features for CNN to learn from. This can lead to the extraction of more diverse and discriminative features, potentially improving the model's overall performance.

## I. Training and Validation Loss and Accuracy

### ***A. Training Loss and Accuracy***

During training, the model learns to minimize a loss function, which measures how well the model's predictions match the actual labels in the training data. The training loss graph shows the value of this loss function over each training epoch (iteration). A decreasing training loss indicates that the model is improving its predictions on the training data.

Similarly, the training accuracy graph shows how the accuracy of the model on the training data changes over each epoch. Accuracy is the proportion of correct predictions made by the model. An increasing training accuracy indicates that the model is getting better at predicting the correct labels on the training data.

## B. Validation Loss and Accuracy

To ensure that the model generalizes well to unseen data, we use a separate validation dataset. The validation loss graph shows the loss on this validation dataset over each epoch. A decreasing validation loss indicates that the model is generalizing well and not overfitting to the training data.

The validation accuracy graph shows the accuracy of the model on the validation dataset over each epoch. Similar to training accuracy, an increasing validation accuracy indicates that the model is performing well on unseen data.

Both graphs, **Figure 9** show that training and validation loss decrease over time, and both training and validation accuracy increase. This indicates that the model is learning from the data and generalizing well. If the training loss continues to decrease while the validation loss starts to increase, it could indicate overfitting. Similarly, if the training accuracy continues to increase while the validation accuracy starts to decrease, it could also indicate overfitting.

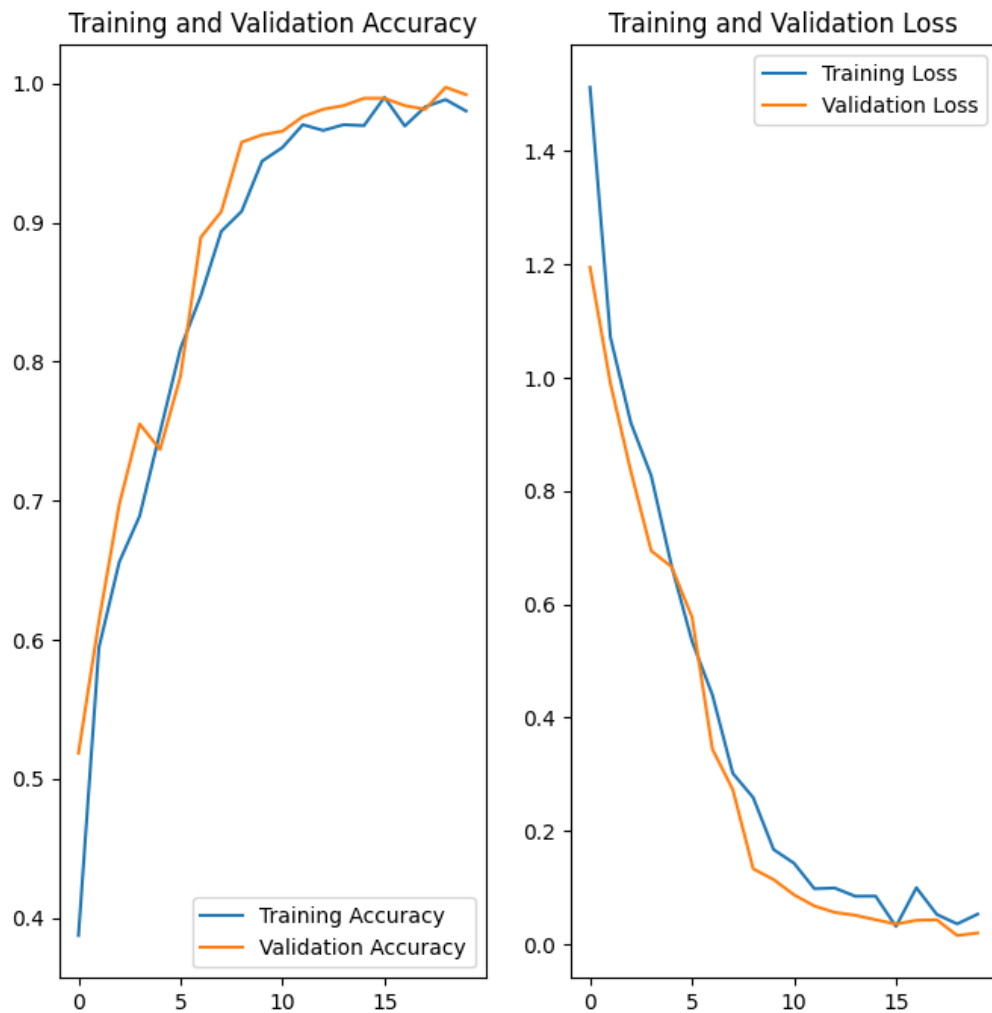


Figure 9

Comparing the models used in other works, as shown in **Table 5**, we can observe the effectiveness of PSO-CNN.

**Table 5**

Number of Diseases	Type of Disease	Methods	Highest Accuracy (%)	Reference
1	Bakanae disease	SVM, GA	87.90	Chung et al. (2016)
10	Rice diseases including rice blast, rice false, smut, rice brown spot, etc.	Novel CNN with 7 layers	95.48	Lu et al. (2017)
9	Rice diseases including false smut, brown plant hopper, etc.	Simple CNN	93.30	Rafeed Rahman et al. (2018)
1	Rice blast	CNN, SVM, LBPH	95.83	Liang et al. (2019)
4	Rice diseases include rice blast, bacterial blight, and sheath rot, brown spot	DNN_JOA, ANN, DAE	97.00	Ramesh and Vydeki (2020)
1	Rice blast	RiceTalk	89.40	Chen et al. (2019)
1	Different levels of contamination of grain discoloration	InceptionV3	88.20	Duong-Trung et al. (2019)
3	Rice blast. bacterial leaf blight, sheath blight	AlexNet CNN + SVM	91.37	Shrivastava et al. (2019)
6	Bacterial Leaf Blight, Brown Spot, Healthy Rice Leaf, Leaf Blast, Leaf scald, and Sheath Blight	PSO-CNN	99.00	Proposed model

This work draws inspiration from Liu et al.'s (2024) work, where they discuss PSOC-DRCNET. The ideas presented are solid; however, there is a lack of documentation. The work mentions Dual Mode Attention (DMA), but there is limited exploration and no accompanying code for reference. Similarly, the use of the Chameleon optimizer is computationally expensive, something we aim to avoid in our work. Nonetheless, we drew inspiration from their explanation of how PSO is advantageous and how it can be leveraged. While many concepts mentioned in the paper are promising, there is little to no documentation available regarding their implementation.

## Evaluation And Reflection

We will evaluate the performance of our Particle Swarm Optimization - Convolutional Neural Network (PSO-CNN) against a normal CNN with random parameters. To ensure a fair evaluation, we will keep all other parameters the same. **Table 6** shows the model summary for the randomized CNN.

**Table 6**

Layers (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 128, 128, 3)	0
conv2d_3 (Conv2D)	(None, 126, 126, 35)	980
max_pooling2d_3 (MaxPooling2D)	(None, 63, 63, 65)	0
conv2d_4 (Conv2D)	(None, 61, 61, 128)	40448
max_pooling2d_4 (MaxPooling2D)	(None, 30, 30, 128)	0
conv2d_5 (Conv2D)	(None, 28, 28, 64)	73792
max_pooling2d_5 (MaxPooling2D)	(None, 14, 14, 64)	0
flatten_1 (Flatten)	(None, 12544)	0
dense_2 (Dense)	(None, 128)	1605760
dense_3 (Dense)	(None, 6)	774
Total params: 5,165,264 (19.70 MB)		
Trainable params: 1,721,754 (6.57 MB)		
Non-trainable params: 0 (0.00 Byte)		

We will compare the classification reports for 10 epochs of both the PSO-CNN and the randomized CNN. **Table 8** shows the classification report for the PSO-CNN, while **Table 7** shows the classification report for the randomized CNN.

**Table 7**

Class	Precision	Recall	F1-Score	Support
Bacterial Leaf Blight	0.98	0.85	0.91	65
Brown Spot	0.90	0.98	0.94	66
Healthy Rice Leaf	0.99	1.00	0.99	70
Leaf Blast	0.94	0.94	0.94	67
Leaf scald	0.93	0.94	0.94	71
Sheath Blight	0.96	0.98	0.97	50
Accuracy			0.95	389
Macro Avg	0.95	0.95	0.95	389
Weighted Avg	0.95	0.95	0.95	389

**Table 8**

<b>Class</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-Score</b>	<b>Support</b>
Bacterial Leaf Blight	1.00	0.92	0.96	59
Brown Spot	0.98	1.00	0.99	81
Healthy Rice Leaf	1.00	1.00	1.00	68
Leaf Blast	0.92	1.00	0.96	61
Leaf scald	0.97	0.94	0.95	77
Sheath Blight	0.98	1.00	0.99	54
Accuracy			0.97	400
Macro Avg	0.98	0.98	0.97	400
Weighted Avg	0.98	0.97	0.97	400

Our analysis indicates that the PSO-CNN model performs slightly better than a randomized CNN model. The true potential of PSO can be further realized by increasing the number of epochs, layers, and hyperparameters. PSO can also be leveraged to optimize other hyperparameters, such as the learning rate.

As we expand the scope of this project beyond rice diseases to include other crops with different types of diseases, the dataset will grow, requiring additional CNN layers and hyperparameters. PSO simplifies this process by enabling us to specify a range for our hyperparameters. For instance, as our dataset expands, we may use multiple neurons. Furthermore, PSO can optimize other hyperparameters, such as the learning rate, which is crucial for model performance.

While this project is focused on rice plants, the same model can be applied to different crops. Combining datasets from various crops can enable the model to perform multi-crop disease detection. However, this approach is computationally expensive and requires additional training layers and time, making it a potential area for future research and development.

## I. Reflection

Reflecting on the project, our main objective was to ensure ease of access for our consumers. We have developed a powerful Android app that, despite its capabilities, is very simple to use. Figure 10 and Figure 11 illustrate this simplicity, where users can simply select "Launch Gallery" or "Take Picture" and wait for the results.

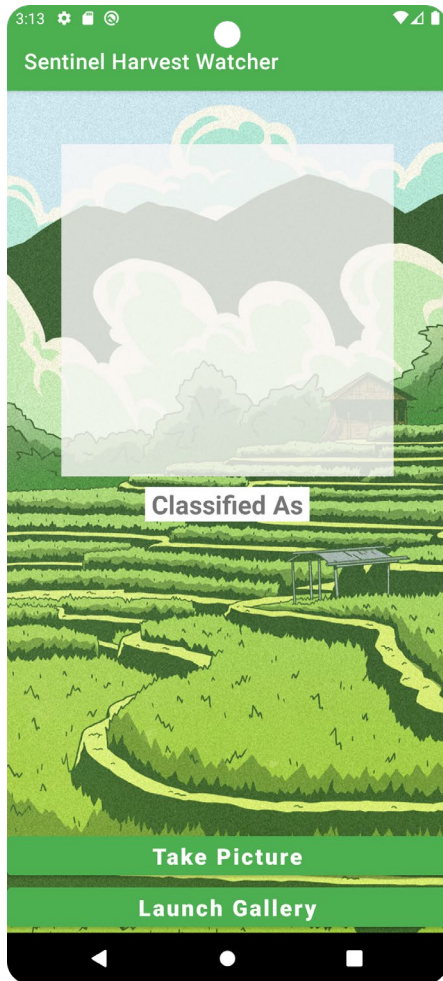


Figure 10



Figure 11

The simplicity of the user interface is crucial as our primary users are farmers. It is remarkable to see how advancements in AI/ML have enabled such complex models to be transformed into simple, one-click applications.



## **II. Social Impact**

### ***A. Improving Crop Yield***

By accurately detecting and identifying diseases in rice plants, farmers can take timely actions to treat affected plants, potentially increasing crop yield and reducing economic losses.

### ***B. Food Security***

Rice is a staple food for a large part of the world's population. By helping farmers identify and manage diseases effectively, this project contributes to ensuring food security for communities that rely on rice as a primary food source.

### ***C. Reducing Pesticide Use***

Early and accurate disease detection can help farmers target treatments only to affected plants, reducing the overall use of pesticides, and minimizing their environmental impact.

### ***D. Empowering Farmers***

Providing farmers with tools and technology to monitor and manage crop diseases can empower them to make more informed decisions, improving their livelihoods and economic stability.

### ***E. Research and Development***

The techniques and methodologies developed in this project can contribute to ongoing research and development in the field of agriculture, advancing our understanding and management of crop diseases.

Overall, the social impact of this project lies in its potential to improve agricultural practices, enhance food security, and promote sustainable farming practices.

## **III. Future Scope**

While this project is currently limited to a rice dataset, its scope can be expanded by adding additional crops. However, this would require refining our CNN model and potentially using other powerful models. As observed, PSO is extremely powerful in determining hyperparameters, and as the dataset grows, so will the layers and additional parameters that need adjustment, such as the learning rate or kernel size, especially if working with models other than CNN.

Another future scope for the project would be to suggest the type of pesticide based on the disease. This would require separate training, data, and model development, but it could provide an optimal solution.

## References

1. K. Ahmed, T. R. Shahidi, S. M. Irfanul Alam and S. Momen, "Rice Leaf Disease Detection Using Machine Learning Techniques," *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*, Dhaka, Bangladesh, 2019, pp. 1-5, doi: 10.1109/STI47673.2019.9068096.
2. Wang, Y., Wang, H., & Peng, Z. (2021). Rice diseases detection and classification using attention based neural network and bayesian optimization. *Expert Systems with Applications*, 178, 114770.
3. Ahad, M. T., Li, Y., Song, B., & Bhuiyan, T. (2023). Comparison of CNN-based deep learning architectures for rice diseases classification. *Artificial Intelligence in Agriculture*, 9, 22-35.
4. Hassan, S. M., & Maji, A. K. (2022). Plant disease identification using a novel convolutional neural network. *IEEE Access*, 10, 5390-5401.
5. Chung, C. L., Huang, K. J., Chen, S. Y., Lai, M. H., Chen, Y. C., & Kuo, Y. F. (2016). Detecting Bakanae disease in rice seedlings by machine vision. *Computers and electronics in agriculture*, 121, 404-411.
6. Lu, Y., Yi, S., Zeng, N., Liu, Y., & Zhang, Y. (2017). Identification of rice diseases using deep convolutional neural networks. *Neurocomputing*, 267, 378-384.
7. Rafeed Rahman, C., Saha Arko, P., Eunus Ali, M., Khan, M. A. I., Hasan Apon, S., Nowrin, F., & Wasif, A. (2018). Identification and Recognition of Rice Diseases and Pests Using Convolutional Neural Networks. *arXiv e-prints*, arXiv-1812.
8. Liang, W. J., Zhang, H., Zhang, G. F., & Cao, H. X. (2019). Rice blast disease recognition using a deep convolutional neural network. *Scientific reports*, 9(1), 1-10.
9. Ramesh, S., & Vydeki, D. (2020). Rice disease detection and classification using deep neural network algorithm. In *Micro-Electronics and Telecommunication Engineering: Proceedings of 3rd ICMETE 2019* (pp. 555-566). Springer Singapore.
10. Duong-Trung, N., Quach, L. D., Nguyen, M. H., & Nguyen, C. N. (2019, January). Classification of grain discoloration via transfer learning and convolutional neural networks. In *Proceedings of the 3rd International Conference on Machine Learning and Soft Computing* (pp. 27-32).

11. Shrivastava, V. K., Pradhan, M. K., Minz, S., & Thakur, M. P. (2019). Rice plant disease classification using transfer learning of deep convolution neural network. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 42, 631-635.
12. Engelbrecht, A. P. (2007). Introduction to computational intelligence. Computational Intelligence, 1-13.
13. Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In Proceedings of ICNN'95- international conference on neural networks (Vol. 4, pp. 1942-1948). ieee.
14. Shi, Y., & Eberhart, R. (1998, May). A modified particle swarm optimizer. In 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360) (pp. 69-73). IEEE.
15. Particle swarm optimization (PSO) with constraint support — pyswarm 0.6 documentation. (n.d.). Pythonhosted.org. <https://pythonhosted.org/pyswarm/>
16. TensorFlow Lite guide. (n.d.). TensorFlow. <https://www.tensorflow.org/lite/guide>
17. Data: Rice Leaf Diseases Detection. (n.d.). Www.kaggle.com. Retrieved May 9, 2024, from <https://www.kaggle.com/datasets/loki4514/rice-leaf-diseases-detection>
18. Introduction — PySwarms 1.3.0 documentation. (n.d.). Pyswarms.readthedocs.io. Retrieved May 9, 2024, from <https://pyswarms.readthedocs.io/en/latest/intro.html#it-s-all-a-treasure-hunt>
19. tf.lite.TFLiteConverter | TensorFlow v2.16.1. (n.d.). TensorFlow. Retrieved May 9, 2024, from [https://www.tensorflow.org/api\\_docs/python/tf/lite/TFLiteConverter](https://www.tensorflow.org/api_docs/python/tf/lite/TFLiteConverter)
20. Raj, A. (2022, June 13). ML in Android -3 : Deploy Tflite on Android using Java. The STEM. <https://medium.com/the-stem/ml-in-android-3-deploy-tflite-on-android-using-java-6a18431644b6>
21. Sunil, C. K., Jaidhar, C. D., & Patil, N. (2023). Tomato plant disease classification using multilevel feature fusion with adaptive channel spatial and pixel attention mechanism. Expert Systems with Applications, 228, 120381.
22. Chang, B., Wang, Y., Zhao, X., Li, G., & Yuan, P. (2024). A general-purpose edge-feature guidance module to enhance vision transformers for plant disease identification. Expert Systems with Applications, 237, 121638.
23. Liu, Z., Zhou, G., Zhu, W., Chai, Y., Li, L., Wang, Y., ... & Sun, L. (2024). Identification of rice disease under complex background based on PSOC-DRCNet. Expert Systems with Applications, 249, 123643.