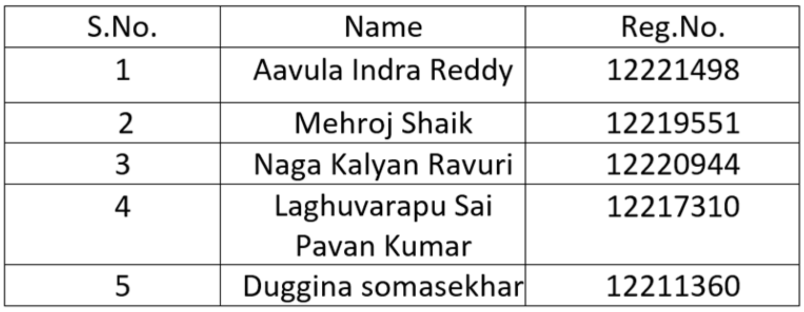
CAPSTONE PROJECT–1 REPORT

An Assignment

Submitted by



S.No.	Name	Reg.No.
1	Aavula Indra Reddy	12221498
2	Mehroj Shaik	12219551
3	Naga Kalyan Ravuri	12220944
4	Laghuvarapu Sai Pavan Kumar	12217310
5	Duggina somasekhar	12211360

Section: KC524

Submitted to:

Ms. Sheveta

B.TECH (COMPUTER SCIENCE ENGINEERING)

Lovely Professional University
Punjab, India

Contents

1 Introduction

Rice is one of the most important food crops in the world. It serves as the main source of nutrition for more than half of the global population. Ensuring healthy rice production is essential for food security and economic stability. However, rice plants are very vulnerable to various leaf diseases, including Bacterial Leaf Blight, Brown Spot, Leaf Blast, Leaf Scald, and Sheath Blight. These diseases can spread quickly, lower crop productivity, and cause significant financial losses for farmers.

Traditionally, farmers or agricultural experts have relied on manual inspection to identify diseases in rice cultivation. However, this method is often limited by several issues:

- Subjectivity, as diagnoses can vary from person to person
- Lack of expertise in remote or rural areas
- Time-consuming visual inspection
- Difficulty in spotting early-stage symptoms

With the rise of Artificial Intelligence and computer vision, new techniques can automate disease detection using images of plant leaves. Among these methods, Convolutional Neural Networks (CNNs) have shown to be very effective at extracting visual features and classifying images accurately.

CNNs replicate the human visual system by learning features in layers. They start with simple details like edges and colors and progress to more complex textures and disease-specific patterns. This ability makes them especially good for diagnosing agricultural diseases, where color changes, lesions, and texture differences are important.

This project aims to build a CNN-based system for the automatic classification of rice leaf diseases. It will use a dataset with thousands of labeled images. The model will learn to analyze the visual patterns linked to different diseases. The goal is to create a system that is accurate, efficient, and reliable. This system can help farmers and agricultural teams identify diseases early and take prompt preventive action.

2 Problem Statement

Rice crops are very vulnerable to various leaf diseases that can greatly lower yield and affect overall crop productivity. Early detection of these diseases is essential for effective crop management. However, manually identifying rice leaf diseases is difficult because:

- It requires expert knowledge and experience.
- Symptoms of different diseases often look similar.
- Large farms make manual inspection time-consuming.
- Human diagnosis can be inconsistent and error-prone.

Because of these challenges, there is an increasing need for a reliable and efficient system that can identify rice leaf diseases from images.

The problem this project addresses is:

"To develop a Convolutional Neural Network (CNN) based model that can automatically classify rice leaf images into different disease categories with high accuracy. This will help farmers and agricultural experts detect diseases early and improve crop management."

3 Dataset Description

The dataset used in this project is essential for the rice leaf disease classification system. It provides the visual information needed for the Convolutional Neural Network (CNN) to learn and accurately distinguish between different disease categories. The dataset consists of rice leaf images collected from actual agricultural fields and research environments. This ensures that the model encounters realistic variations found in practical situations. These variations include differences in lighting, leaf orientation, disease intensity, background settings, and image resolution. Such diversity helps the model generalize better and perform accurately in real-world farming applications.

The dataset is sourced from Kaggle and includes six main classes of rice leaf conditions. These classes represent some of the most common and damaging diseases affecting rice crops. They are:

- Bacterial Leaf Blight
- Brown Spot
- Healthy Rice Leaf
- Leaf Blast
- Leaf Scald
- Sheath Blight

Each class contains several hundred to thousands of images, ensuring balanced coverage to prevent model bias. The representation of both diseased and healthy leaves is important, as it allows the model to differentiate between normal leaf patterns and visually distinct symptoms of diseases. Conditions such as Leaf Blast and Sheath Blight show clear symptoms like lesions, discoloration, texture changes, and irregular patches. The CNN learns to identify these patterns through repeated feature extraction during training.

Here are sample images representing the different rice leaf conditions:

3.1 Dataset Classes

The dataset is divided into six classes:

- **Bacterial Leaf Blight:** This class includes yellowing and wilting of leaves caused by bacterial infection.
- **Brown Spot:** This class features brown circular lesions with clear borders.
- **Healthy Rice Leaf:** This class consists of normal green leaves without spots, lesions, or discoloration.

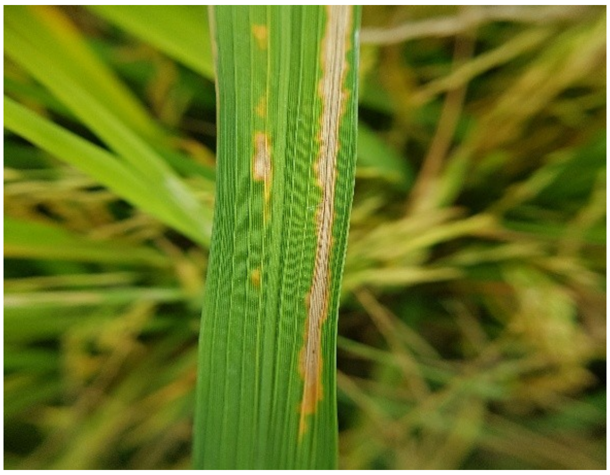


Figure 1: Bacterial Leaf Blight

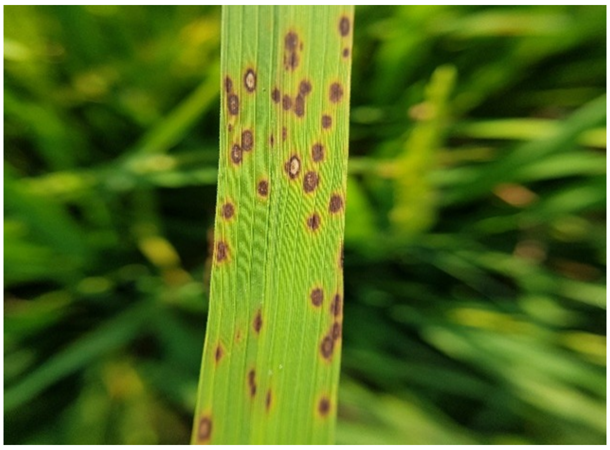


Figure 2: Brown Spot

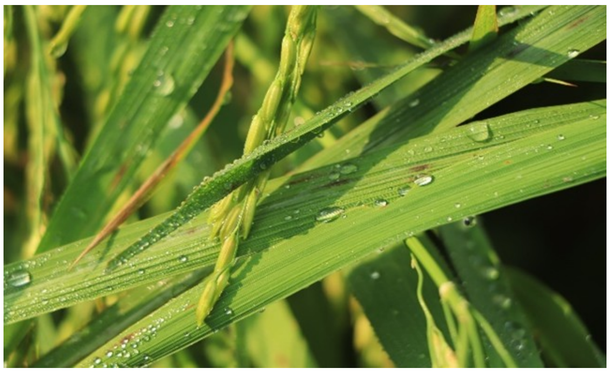


Figure 3: Healthy Rice Leaf

- **Leaf Blast:** This class has spindle-shaped lesions with grey centers and brown edges.
- **Leaf Scald:** This class shows significant dead areas and burnt-like patches on the leaves.
- **Sheath Blight:** This class includes oval lesions and oddly shaped fungal patches, often spreading along the leaf sheath.

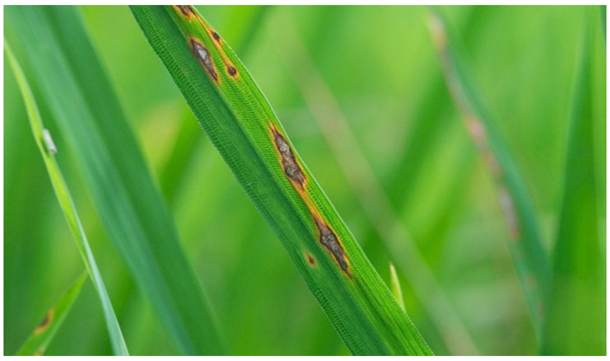


Figure 4: Leaf Blast

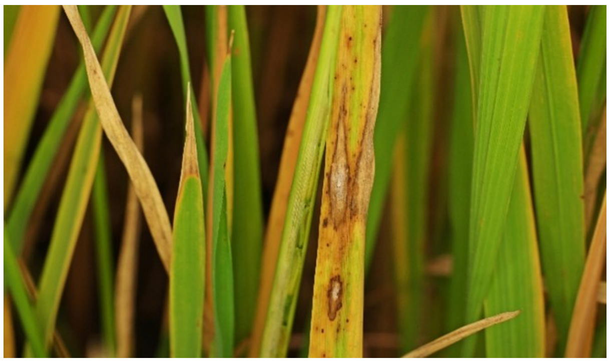


Figure 5: Leaf Scald

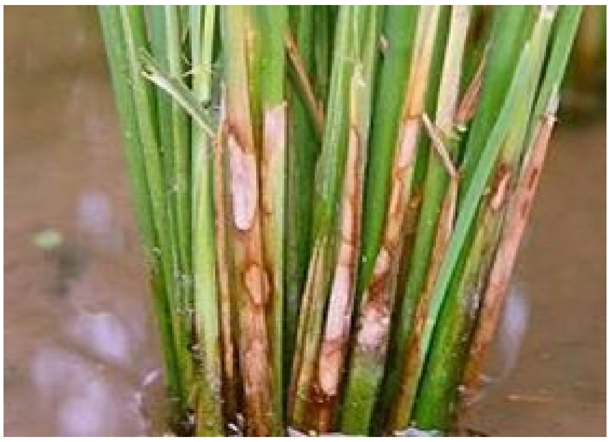


Figure 6: Sheath Blight

These distinct visual features help the CNN model recognize the differences between the disease patterns.

3.2 Image Characteristics

The images in the dataset vary significantly in quality and environmental context. They come from different cameras and were taken under various conditions. This variability enhances the model's strength by ensuring it can understand images captured in both controlled and uncontrolled settings.

Key characteristics include:

- Different lighting conditions (bright sunlight, shade, dim light)
- Multiple leaf angles and orientations
- Varying disease intensities (early-stage, mid-stage, advanced)
- Varied backgrounds such as:
  - Soil
  - Field grass
  - Farmer's hand
  - Concrete floor
  - Other plants or tools
- Variations in resolution: high-resolution images with clear details and low-resolution images that may be blurry or noisy

These natural variations ensure the model does not overfit on uniform backgrounds and lighting.

3.3 Dataset Size and Distribution

The dataset underwent additional processing and organization for consistency. It was divided into three subsets:

- 70% Training Data
- 15% Validation Data
- 15% Test Data

This distribution ensures:

- The model learns from a large portion of the data
- Validation helps in tuning hyperparameters
- Testing measures real-world performance

Example distribution (numbers may vary slightly):

- Training Images: $\sim$2600
- Validation Images: $\sim$570
- Testing Images: $\sim$580

The balanced distribution supports fair learning and evaluation without data leakage.

3.4 Dataset Cleaning and Renaming

Before preprocessing, the dataset included folder names with spaces, uppercase letters, or inconsistencies, such as:

- "Bacterial Leaf Blight"
- "Healthy Rice Leaf"
- "Leaf scald"

These were standardized to machine-friendly formats:

- `bacterial_leaf_blight`
- `healthy_rice_leaf`
- `leaf_scald`

Standardizing folder names is important for automated dataset loading using libraries such as TensorFlow and Keras.

4 Image Preprocessing Applied to the Dataset

To prepare images for training, the following preprocessing steps were applied:

1. Image Resizing

All images were resized to:

$$224 \times 224 \times 3$$

This ensures a uniform input size for the CNN.

2. Normalization

Pixel values were scaled from the range 0–255 to 0–1 to help the model train efficiently.

3. Data Augmentation

To increase dataset diversity and prevent overfitting, the following augmentation techniques were applied:

- Rotation
- Horizontal flip
- Random zoom
- Shear transformation
- Brightness variation

4. Data Shuffling

This ensured that images fed to the model were randomly selected, preventing learning bias.

4.1 Importance of This Dataset

The dataset is essential for:

- Teaching the CNN how different rice diseases appear
- Understanding subtle visual differences between leaf conditions
- Achieving high accuracy through diverse and realistic image samples
- Developing a deployable solution for farmers and agricultural experts

Because the dataset covers real-world image variations, the trained model becomes more reliable and supports precision agriculture by reducing pesticide misuse.

5 Working of CNN

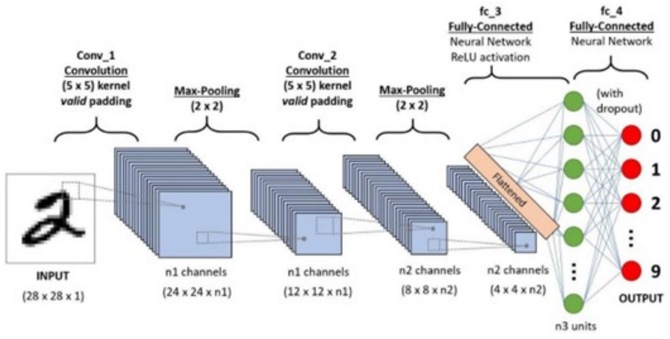


Figure 7: CNN Architecture

5.1 Convolution Operation

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can process an input image, assign importance to different aspects or objects in the image, and distinguish one from another. The pre-processing needed for a ConvNet is much lower compared to other classification algorithms. In older methods, filters are manually created, but with enough training, ConvNets can learn these filters and characteristics.

The structure of a ConvNet is similar to how neurons connect in the human brain. It was inspired by the layout of the visual cortex. Individual neurons respond to stimuli only in a limited area of the visual field called the receptive field. A group of these fields overlaps to cover the whole visual area. ConvNets are a type of artificial neural network (ANN) mainly used to analyze visual imagery. They are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN), based on the shared-weight design of the convolution kernels or filters that move along features and produce translate-equivariant responses called feature maps. Interestingly, most convolutional neural networks are only equivariant, rather than invariant, to translation.

ConvNets have various applications including image and video recognition, recommendation systems, image classification, image segmentation, medical systems, image analysis, natural language processing, brain-computer interfaces, and financial time series. CNNs are regularized versions of multi-layer perceptrons. Multi-layer perceptrons typically refer to fully connected networks, meaning each neuron in one layer connects to all neurons in the next layer. This full connectivity makes them more likely to overfit data. Common methods to reduce overfitting include penalizing parameters during training, like weight decay, or reducing connectivity through methods like skipped connections or dropout.

CNNs use a different method for regularization. They leverage the hierarchical nature of data and build patterns of increasing complexity using smaller and simpler patterns in their filters. Therefore, in terms of connectivity and complexity, CNNs are on the simpler side. Convolutional networks draw inspiration from biological processes, with their connectivity patterns resembling the organization of the animal visual cortex. Individual cortical neurons respond only within their receptive fields, which partially overlap to cover the entire visual field. CNNs use relatively little pre-processing compared to other image classification methods.

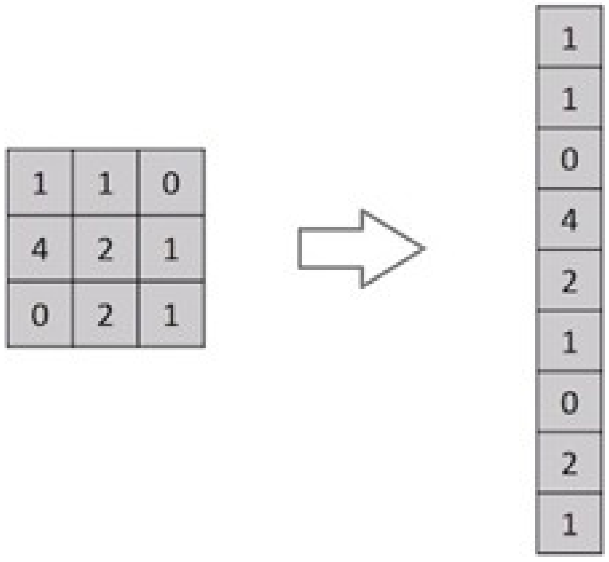


Figure 8: Flattening of a 3x3 image matrix into a 9x1 vector
An image is nothing but a matrix of pixel values. So flatten the image(e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification. In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout. A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

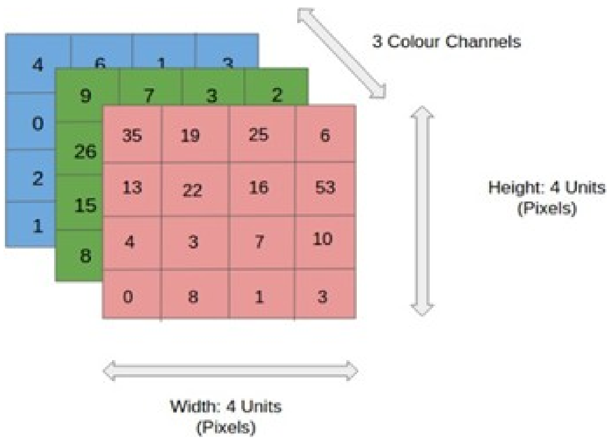


Figure 9: 4x4x3 RGB Image

In the above figure, we have an RGB image which has been separated by its three colour planes — Red, Green, and Blue. There are a number of such colour spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc. You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

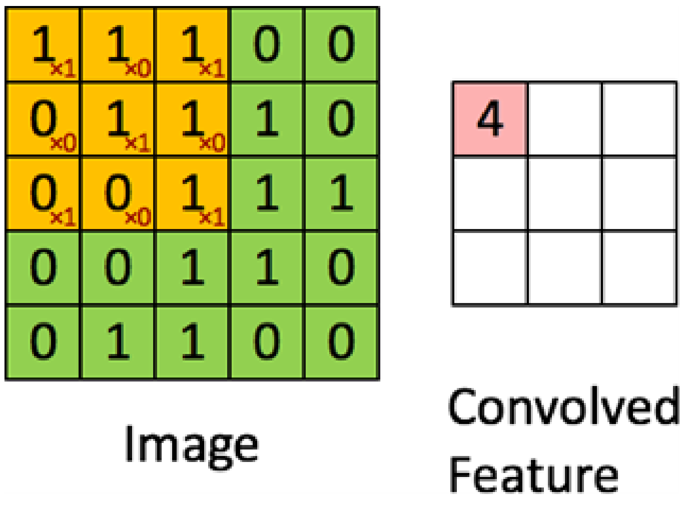


Figure 10: Convoluting a 5x5x1 image with a 3x3x1 kernel

Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB).In the above demonstration, the green section resembles our 5x5x1 input image, I. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the color yellow. We have selected K as a 3x3x1 matrix. The Kernel shifts 9 times because of Stride Length = 1 (Non-Strided), every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering.

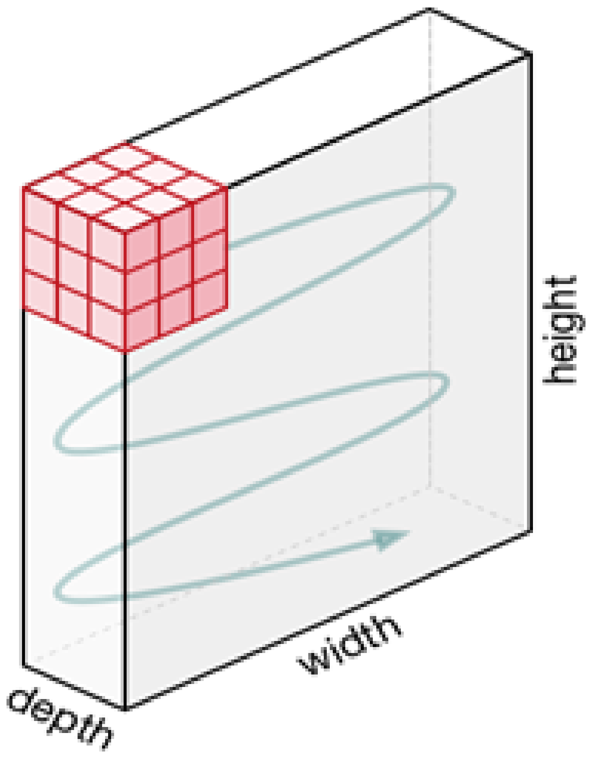


Figure 11: Movement of the kernel

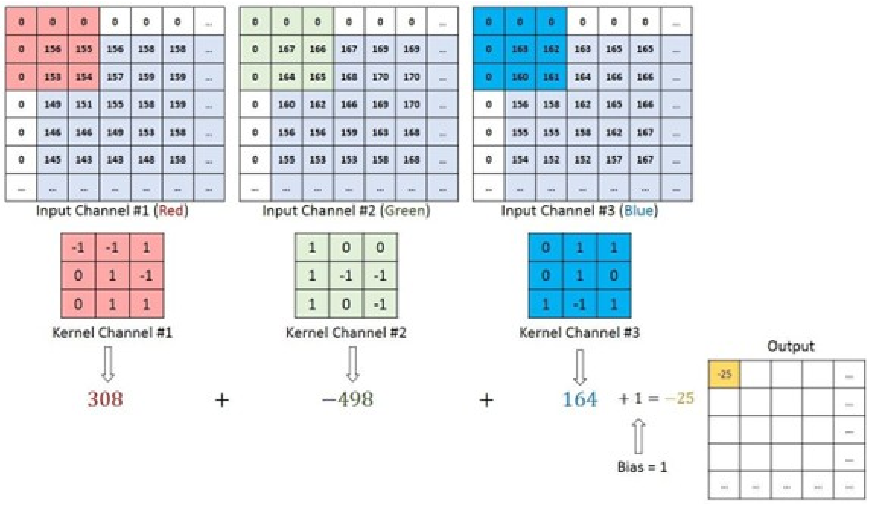


Figure 12: Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

5.2 Padding

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between Kn and In stack ([K1, I1]; [K2, I2]; [K3, I3]) and all the results are summed with the bias to give us a squashed one-depth channel Convoluted Feature Output.

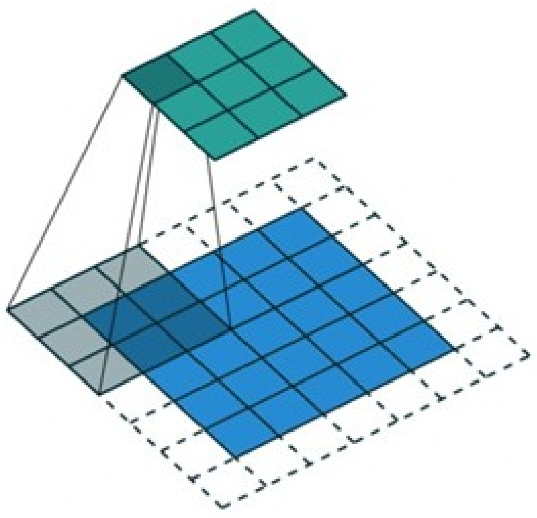


Figure 13: Convolution Operation with Stride Length = 2

5.3 Same Padding

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, similar to how we would. There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying Valid Padding in case of the former, or Same Padding in the case of the latter.

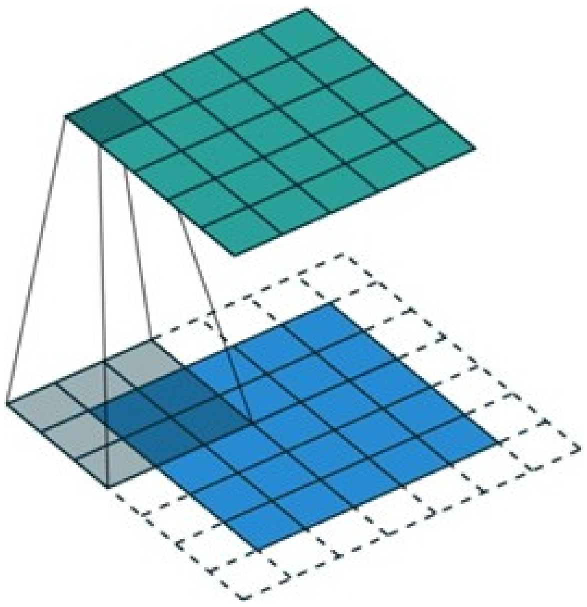


Figure 14: 5x5x1 image is padded with 0s to create a 6x6x1 image

When we augment the 5x5x1 image into a 6x6x1 image and then apply the 3x3x1 kernel over it, we find that the convolved matrix turns out to be of dimensions 5x5x1. Hence the name — Same Padding. On the other hand, if we perform the same operation without padding, we are presented with a matrix which has dimensions of the Kernel (3x3x1) itself — Valid Padding. The following repository houses many such GIFs which would help you get a better understanding of how Padding and Stride Length work together to achieve results relevant to our needs.

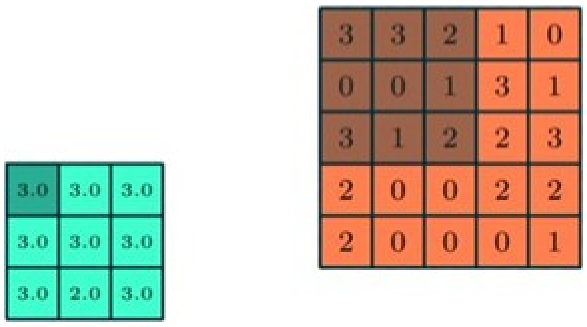


Figure 15: 3x3 polling over 5x5 convolved feature

5.4 Pooling

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model. There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de- noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

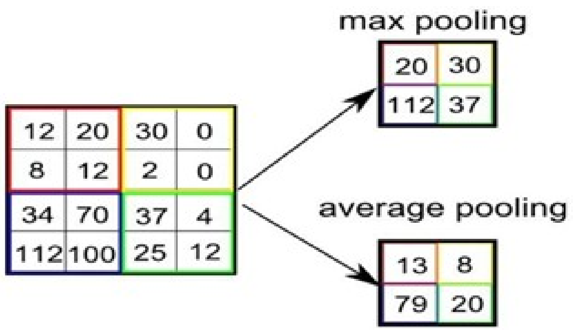


Figure 16: Types of Pooling

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power. After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

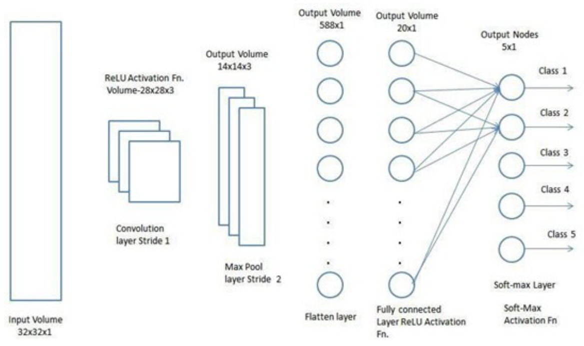


Figure 17: Classification-Fully Connected layer

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space. Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the

6 Methodology

6.1 Dataset Loading and Organization

The first step was to obtain the rice leaf disease dataset from Kaggle. This dataset includes images from six different categories of rice leaf conditions. After downloading, the dataset was moved into the working directory, and the class folders were reviewed. Some folder names contained spaces or inconsistent formatting, which could cause errors when loading data with deep learning libraries. Therefore, the folder names were cleaned and standardized into lowercase, underscore-separated labels such as:

- bacterial_leaf_blight
- brown_spot
- healthy_rice_leaf
- leaf_blast
- leaf_scald
- sheath_blight

This ensured compatibility with TensorFlow's `flow_from_directory()` function and made dataset handling easier.

6.2 Data Splitting

To train and evaluate the model properly, the dataset was divided into three subsets:

- 70% Training Data
- 15% Validation Data
- 15% Testing Data

This split ensures that:

- The model learns from a large portion of the data
- Validation helps in monitoring performance during training
- Testing evaluates the model on unseen data

The splitting was performed using Python scripts, ensuring each subset included images from all six classes without duplication.

6.3 Image Preprocessing

Before inputting images into the CNN, several preprocessing steps were applied to standardize the data and improve model performance.

Image Resizing All images were resized to $224 \times 224 \times 3$ to meet the CNN input requirements.

Normalization Pixel values were scaled from 0–255 to 0–1 using:

Data Augmentation To increase dataset diversity and prevent overfitting, real-time augmentation techniques were applied using `ImageDataGenerator`, including:

- Random rotation
- Horizontal flipping
- Zoom transformation
- Shear operations

These variations help the model learn more generalized and robust features.

6.4 Building the CNN Model

The CNN architecture was custom-designed to extract disease features at multiple levels. The model consists of:

- 4 Convolutional blocks with increasing filters (32, 64, 128, 256)
- MaxPooling layers after each Conv2D layer
- Global Average Pooling to reduce feature map dimensions
- Dense layer with 256 neurons for feature interpretation
- Dropout layer (40%) to avoid overfitting
- Final Softmax layer with 6 neurons for classification

This architecture balances simplicity, performance, and efficiency.

6.5 Compiling the Model

The model was compiled using:

- **Loss Function:** Categorical Crossentropy
- **Optimizer:** Adam
- **Metrics:** Accuracy

The Adam optimizer was chosen due to its fast convergence and adaptive learning rate capabilities.

6.6 Training the Model

The training process involved passing batches of images through the CNN and adjusting the parameters via backpropagation. Training was conducted over multiple epochs, allowing the model to gradually improve its ability to classify diseases.

During training:

- Training accuracy and loss were continuously monitored
- Validation accuracy and loss helped detect overfitting
- Data augmentation introduced variability into training batches
- The model progressively improved its classification performance

6.7 Evaluation on Test Set

After training, the model was evaluated using the test dataset, which contains unseen images. This step provides an unbiased measurement of how well the model generalizes to new data. The final test accuracy and loss were recorded and compared with training and validation metrics.

6.8 Training Visualization

To better understand the learning behavior of the model, two essential graphs were plotted:

- **Accuracy vs Epochs:** Shows changes in training and validation accuracy over time.
- **Loss vs Epochs:** Shows improvements or fluctuations in training and validation loss.

These visualizations help analyze:

- Learning stability
- Convergence patterns
- Signs of overfitting or underfitting
- Overall model performance trends

7 Results and Evaluation

7.1 Final Test Accuracy

After training the CNN model for 20 epochs, the evaluation on the test dataset produced the following results:

- **Final Test Accuracy:** 83.88%
- **Final Test Loss:** 0.4625

These values indicate that the model performs well on unseen images and generalizes effectively across the six rice leaf disease classes.

7.2 Training and Validation Performance

Throughout the training process, both accuracy and loss showed steady improvement. Key observations include:

- Training accuracy improved from 65.59% (Epoch 1) to 82.32% (Epoch 20).
- Validation accuracy increased from 69.51% to 84.32%, demonstrating strong generalization.
- Validation loss steadily decreased, reaching 0.5148 at the final epoch.

The closeness between the training and validation accuracy curves indicates minimal overfitting.

7.3 Accuracy Curve

- Training accuracy showed a consistent upward trend over the epochs.
- Validation accuracy improved steadily and peaked at 84.32%.
- The convergence between training and validation accuracy reflects stable learning behavior.

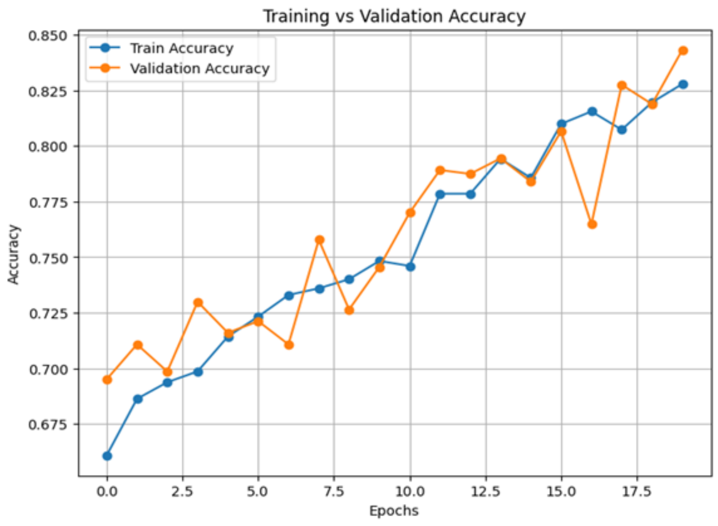


Figure 18

7.4 Loss Curve

- Training loss decreased significantly from 0.9090 to 0.4764.
- Validation loss reduced from 0.7796 to 0.4928.
- No major fluctuations or instability were observed, indicating smooth optimization during training.

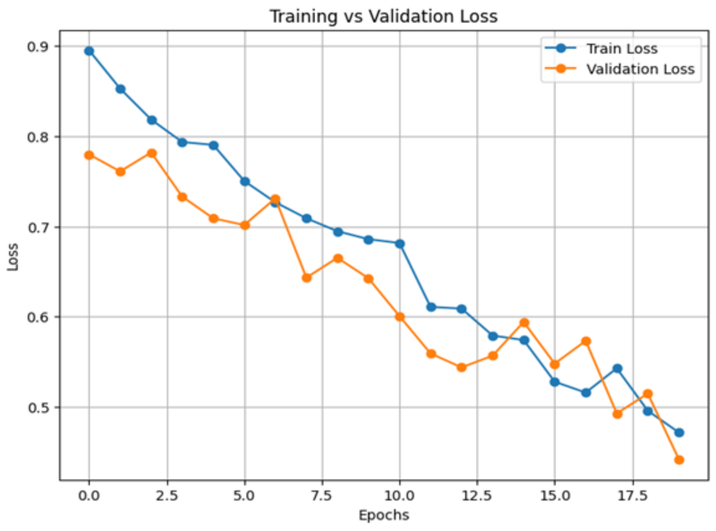


Figure 19

8 Conclusion

This project successfully developed a Convolutional Neural Network (CNN) model that accurately classifies six major rice leaf diseases. By using systematic preprocessing, data augmentation, and a carefully designed CNN architecture, the model achieved a strong final test accuracy of 83.88The results indicate that deep learning, especially CNN-based architectures, can serve as a dependable tool for automated crop disease diagnosis. These systems can help farmers and agricultural professionals detect problems early, reducing crop losses and improving yield quality. Overall, the model provides a solid foundation for future improvements and practical use in agricultural monitoring applications.

9 Future Scope

he current CNN model performs well, but there is significant potential for further improvement and real-world deployment. Future work can look into integrating pretrained deep learning models like EfficientNet, MobileNetV2, ResNet50, and InceptionV3, known for their excellent feature-extracting capabilities and efficiency. These architectures can be fine-tuned on the rice disease dataset to achieve higher accuracy, faster convergence, and better generalization. Additionally, the model can be expanded by incorporating:

- Larger and more diverse datasets collected in real field conditions
- Improved augmentation techniques to simulate lighting, noise, and environmental variations
- Attention-based models, like Vision Transformers, for better focus on infected areas
- Deployment on mobile devices to create real-time disease detection tools for farmers
- Integration with IoT and drone systems for large-scale crop health monitoring

By using advanced pretrained models and modern AI techniques, this system can evolve into a high-precision agricultural diagnostic tool suited for real-world use.

10 References

1. Ahmed, K., Shahidi, T.R., Irfanul Alam, S.M., Momen, S. (2019). Rice leaf disease detection using Machine Learning techniques. *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI).* IEEE, Dhaka, Bangladesh.

2. Taskeen Ashraf, Yasir Niaz Khan. (2020). Weed density classification in rice crop using computer vision. *Computers and Electronics in Agriculture*, 175.

3. Sanga, S.L., Machuve, D., Jomanga, K. (2020). Mobile-based deep learning models for Banana Disease detection. *Technology and Applied Sciences Research*, 10(3).

4. Tewari, V.K., Pareek, C.M., Lal, G., Dhruv, L.K., Singh, N. (2020). Image processing based real-time variable-rate chemical spraying system for disease control in paddy. *Artificial Intelligence in Agriculture.*

5. Ramesh, S., Vydeki, D. (2020). Recognition and classification of paddy leaf diseases using optimized deep neural network with Jaya algorithm. *Information Processing in Agriculture*.

6. Sethy, P.K., Barpanda, N.K., Rath, A.K., Behera, S.K. (2020). Deep feature-based rice leaf diseases identification using support vector machine. *Computers and Electronics in Agriculture*.

7. Li, Y., Yang, J. (2020). Few-shot cotton pest recognition and terminal realization. *Computers and Electronics in Agriculture*, 169, 105240.

8. Lee, S.H., Goeau, H., Bonnet, P., Joly, A. (2020). New perspectives on plant disease characterization based on deep learning. *Computers and Electronics in Agriculture*, 170, 105220.